Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Trabajo Fin de Grado

Aplicación móvil para el control domótico de sensores instalados en una vivienda

Autor: Leandro Santiago Otón González

Tutor: Bernardo Alarcos Alcázar

2016

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Trabajo Fin de Grado

Aplicación móvil para el control domótico de sensores instalados en una vivienda

Autor: Leandro Santiago Otón González

Tutor: Bernardo Alarcos Alcázar

TRIBUNAL:

Presidente: Miguel Ángel López Carmona

Vocal 1º: Juan Ramón Velasco Pérez

Vocal 2º: Bernardo Alarcos Alcázar

CALIFICACIÓN: .....

FECHA: .....

Agradecimientos

Quiero agradecer a mi padre, Leandro, a mi madre, Puri, a mi hermana, Lucía y a mi hermano, Luismi, todo el apoyo brindado durante estos años de mi vida. Sin ellos ninguno de mis sueños podría haberse cumplido.

Quiero agradecer a mi novia, Arancha, por estar a mi lado apoyando cada una de mis decisiones.

Y en definitiva, agradecer a todos mis amigos, compañeros de universidad y profesores que siempre confiaron en mí.

Leandro Santiago Otón González

Índice

Índice de figuras y tablas

Índice de código

Resumen

Se propone el desarrollo de una aplicación de domótica para dispositivos móviles que permita gestionar los sensores de una vivienda, comunicándose con la base de datos de ThingSpeak mediante peticiones HTTP que serán interpretadas y resueltas por la propia REST API de ThingSpeak. Al mismo tiempo se emularán, mediante un desarrollo web, los sensores cambiando de valor y su actualizándose en la base de datos.

Se hace uso de HTML5, CSS3 y JAVASCRIPT, junto con JQUERY, para el desarrollo de la aplicación y de la herramienta INTEL XDK para su compilación para diversas plataformas móviles (iOS, Android, WindowsPhone).

Palabras Clave

Aplicación de domótica, gestión de sensores, peticiones HTTP, REST API, dispositivos móviles iOS-Android-WindowsPhone.

Abstract

A domotic application for mobile phones is developed in this Project. It allows managing home sensors communicating with the database from ThingSpeak via HTTP requests. These will be interpreted and resolved by the ThingSpeak REST API. At the same time the sensors will be emulated, via web deployment, to update the database with new values.

HTML5, CSS3 and JAVASCRIPT, plus JQUERY, are used for the application deployment and the INTEL XDK tool is used to compile the application for Android, iOS and WindowsPhone mobiles phones.

Keywords

Domotic application, managing sensors, HTTP requests, REST API, Android-iOS-WindowsPhone mobiles phones.

Resumen extendido

Este trabajo se centra en el desarrollo de una aplicación domótica para su uso en smartphones, la cual se ha programado haciendo uso de HTML5, CSS3 y JAVASCRIPT, y compilado utilizando la herramienta INTEL XDK proporcionada por Intel la cual nos permite generar aplicaciones HTML5 para Android, iOS, Windows 10, Windows 8 y WindowsPhone. Así mismo, la aplicación pretende que se puedan configurar, gestionar y personalizar de una forma sencilla los diversos interruptores de luz, reguladores de luz, sensores de intensidad luminosa, sensores de temperatura, sensores de humedad y sensores de lluvia que se encuentren en una vivienda y que, además, serán almacenados en la base de datos en la nube de ThingSpeak.

El sistema End-To-End que se ha simulado consta, por un lado, de una aplicación móvil que se comunica con una base de datos (que se encuentra en el entorno de ThingSpeak) mediante peticiones HTTP haciendo uso de una REST API (también propia del entorno de ThingSpeak) que negocia con dicha base de datos. Por otro lado, mediante un entorno web, se ha simulado a una Rasperry que al mismo tiempo que nos permite recoger diferentes valores proporcionados por un conjunto de sensores, también nos permite conectar con la base de datos mediante peticiones HTTP haciendo uso de la REST API de ThingSpeak. De esta forma conseguimos que la persona que utiliza la aplicación pueda leer los datos proporcionados por los interruptores de luz, los reguladores de luz y los sensores.

Las principales características que posee la aplicación son:

1. Interfaz sencilla y agradable a la vista.

2. Fácil configuración y uso de la aplicación, en la cual sólo se deben introducir tres parámetros para que quede configurada y pueda comenzar a ser utilizada.

3. Modificación del nombre de cada campo, disponible para cada tipo de sensor, en la propia aplicación.

La principal aportación que ofrece esta aplicación respecto al resto es que todo el mundo pueda descargarse y disponer de una aplicación que pueda ser configurada y utilizada de una forma muy sencilla y que pueda ser utilizada con cualquier tipo de marca de sensor. Gran parte de las aplicaciones ofertadas en los Markets se centran exclusivamente en su uso para tecnologías concretas como puede ser Raspberry, Arduino, tecnología de Samsung, de LG, etc. Mientras que la aplicación desarrollada en este trabajo, “Home

Smart Home", permite su uso para cualquier dispositivo que sea capaz de actualizar los valores de la base de datos de ThingSpeak.

Abreviaturas

HTTP: Hiper Text Transfer Protocol

REST API: REpresentational State Transfer Application Programming Interface

1. Memoria del proyecto

1.1 Introducción

Este trabajo se centra en el desarrollo de una aplicación de domótica para smartphones capaz de gestionar una serie de sensores cuyo estado de sus valores se encuentra almacenado en la base de datos de ThingSpeak. Los posibles agentes encargados de actualizar el valor de los sensores almacenados en la base de datos es pueden ser un microcontrolador (Arduino) o una computadora funcional (Raspberry). Dichos agentes se encargan de recoger los datos proporcionados por los sensores para su posterior actualización sobre la base de datos. Por otra parte, el desarrollo de dicha aplicación hace uso de los lenguajes HTML5, CSS3 y JAVASCRIPT, mientras que su compilación es realizada mediante una herramienta llamada INTEL XDK.

ThingSpeak es una plataforma abierta creada para el Internet de las Cosas (IoT: Internet of Things) que mediante una REST API permite interactuar de una forma sencilla con una base de datos donde se almacenan y recuperan datos mediante el envío peticiones y respuestas HTTP a través de Internet.

REST API es un servicio que nos provee de funciones que nos dan la capacidad de hacer uso de un servicio web, dentro de una aplicación propia, de manera segura. Para comunicarnos con dicha REST API utilizamos peticiones HTTP.

Las peticiones HTTP son un protocolo orientado a transacciones y siguen el esquema petición-respuesta entre un cliente y un servidor. Los métodos de petición utilizados en esta aplicación son el método GET y el método POST.

INTEL XDK es una herramienta utilizada para desarrollar aplicaciones multiplataforma utilizando tecnologías estándar tales como HTML5, CSS3 y JAVASCRIPT.

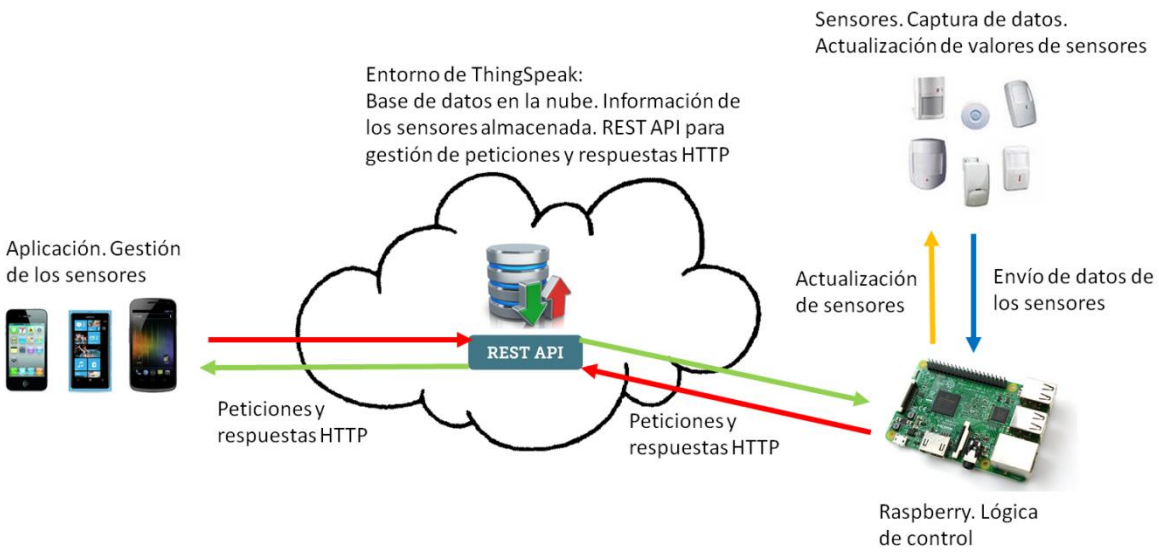

Figura 1.1. Esquema del sistema End-to-End

El sistema End-To-End sobre el que vamos a trabajar (Figura 1.1) consta de una aplicación móvil que se comunica con la una base de datos de ThingSpeak mediante peticiones HTTP que son interpretadas y resueltas por la propia REST API de ThingSpeak. Al mismo tiempo, habrá una Raspberry que también mediante peticiones HTTP se comunica con la base de datos permitiendo la actualización sobre la base de datos del valor de los sensores recogido por la Raspberry.

Dentro del sistema End-To-End descrito anteriormente el objetivo del trabajo en el que nos vamos a centrar consiste en desarrollar una aplicación domótica para Smartphones mediante el uso de HTML5, CSS3 y JAVASCRIPT (haciendo uso de librerías de JQUERY), en el entorno de desarrollo, y la herramienta INTEL XDK, en el entorno de compilación, que a la vez que se comunica con la base de datos en la nube de www.thingspeak.com haciendo uso de la propia REST API de ThingSpeak mediante peticiones HTTP, también permita configurar, gestionar y personalizar en nuestra aplicación de una forma sencilla los diversos interruptores de luz, reguladores de luz, sensores de intensidad luminosa, sensores de temperatura, sensores de humedad y sensores de lluvia que podamos tener en nuestra vivienda y cuyos valores están almacenados en la base de datos de ThingSpeak. El nuevo código de control de los sensores está implementado de tal manera que la persona que utilice la aplicación tan solo tenga que hacer uso del valor de tres parámetros (Channel ID, Read Api Keys y Write Api Key), obtenidos por el usuario al registrarse en la plataforma de ThingSpeak, para de esta forma

generar en el cliente de forma automática los diversos campos de los sensores mencionados anteriormente los cuales previamente habrán sido configurados en la página de www.thingspeak.com por parte de la persona que vaya a utilizar la aplicación. De esta forma cada persona podrá personalizar cuántos sensores desea de cada tipo y al mismo tiempo gestionarlos desde la aplicación.

Para emular la parte de la comunicación formada por los sensores, la Raspberry y la base de datos se ha desarrollado un entorno web muy sencillo con HTML5 y JAVASCRIPT (haciendo uso de librerías de JQUERY) en el que podemos observar el comportamiento de los sensores. Además desde este entorno se pueden modificar los valores de la base de datos para que podamos ver su cambio en la aplicación móvil.

La estructura de los distintos apartados del capítulo es la siguiente:

- 1.2. Base teórica. Se explica la base teórica en que nos hemos basado a la hora de crear la aplicación. Es la motivación del proyecto en sí.

- 1.3. Descripción experimental. Se describe la estructura general del sistema End-to-End: la aplicación, la base de datos, la Raspberry y los sensores.

- 1.4. Entorno de ThingSpeak. Se describe tanto el entorno web proporcionado por ThingSpeak como las funcionalidades de la REST API proporcionada por ThingSpeak para la comunicación con su base de datos, permitiéndonos realizar los desarrollos de la aplicación móvil (1.6 Entorno de la aplicación móvil) y del entorno web (1.7 Entorno del desarrollo web para emular la comunicación entre sensores,Raspberry y base de datos).

- 1.5. Entorno de compilación mediante la herramienta INTEL XKD. Se describe el uso y las funcionalidades que aporta la herramienta de compilación, llamada INTEL XDK, utilizada para generar la aplicación para los smartphones.

- 1.6. Entorno de la aplicación móvil. Se describe tanto el entorno gráfico de la aplicación y sus funcionalidades como la implementación de dichas funcionalidades.

- 1.7. Entorno del desarrollo web para emular la comunicación entre sensores, Raspberry y base de datos. Se describe tanto el entorno gráfico del desarrollo web y sus funcionalidades como la implementación de dichas funcionalidades.

- 1.8. Versión entregada. Se describe para qué tipo de plataforma se ha compilado la aplicación.

- 1.9. Conclusiones y trabajo futuro. Se describen las conclusiones a las que se ha llegado después de la realización del proyecto, junto a los trabajos futuros a realizar en las siguientes versiones.

1.2 Base teórica

El proyecto se basa en la premisa de cualquier persona pueda domotizar y gestionar mediante un smartphone su casa de una forma económica y sencilla. Debido al incremento hoy día del uso de las nuevas tecnologías y en la mayoría de sus casos a su elevado precio, es un requisito indispensable para una persona que el uso de una nueva tecnología que le facilite la vida y le suponga ahorros en su día a día tenga un bajo coste. Por otro lado, no todas las personas tienen las mismas capacidades de intuición y compresión por lo que cuanto más sencilla e intuitiva sea dicha tecnología mejor acogimiento tendrá entre la población. Por último, uno de los mejores usos que se puede dar a las nuevas tecnologías es el de domotizar una vivienda ya que aporta múltiples ventajas tales como el ahorro energético, control y gestión de una forma centralizada de todos los dispositivos de una vivienda, seguridad mediante avisos de alarmas y ayuda para personas mayores y discapacitados para el control de su vivienda. Todo ello se traduce en un nivel de calidad de vida mayor.

Por todo ello se ha desarrollado una aplicación móvil que se conecte con una base de datos en la nube para la gestión domótica de una casa de una forma simple e intuitiva. Dicha aplicación no tiene coste alguno, por lo que su usuario únicamente tendrá que invertir tanto en los sensores que desee instalar en su vivienda como en una Raspberry que haga de receptor de información de los sensores para su posterior envío a la base de datos que se encuentra en la nube.

1.3 Descripción experimental

La estructura general del sistema End-to-End (Figura 1.1) se compone de sensores que recogen datos, de una Raspberry que transfiere esos datos recogidos por los sensores a una base de datos en la nube, la base de datos de ThingSpeak donde se almacenan los datos enviados por la Raspberry y la aplicación móvil que nos permite visualizar el estado de los sensores cuyos valores han sido almacenados en la base de datos y, además, nos permite modificar algunos de esos valores.

El sistema formado por los sensores y la Raspberry es emulado por un desarrollo web que mediante peticiones HTTP se encarga tanto de modificar constantemente los valores de los sensores almacenados en la base de datos como de leer valores que hayan sido

modificados en la base de datos por la aplicación móvil para posteriormente presentarlos sobre el entorno web.

La base de datos en la nube de ThingSpeak es donde se encuentran almacenados los datos de los sensores. A dicha base de datos se accede mediante peticiones HTTP a la REST API que ofrece el entorno de ThingSpeak. Dicha REST API se encarga de gestionar las peticiones y respuestas HTTP sobre la base de datos.

La aplicación móvil se basa en la extracción e inclusión de información sobre la base de datos de ThingSpeak mediante peticiones HTTP que son resueltas por la propia REST API de ThingSpeak. La información obtenida por la aplicación es tratada y representada gráficamente en el terminal permitiéndole al usuario conocer en todo momento el estado de los sensores de su vivienda. Para configurar la aplicación se han de usar tres parámetros proporcionados en la web de ThingSpeak al realizar el registro y creación de un nuevo canal. Estos tres parámetros son: Channel ID, Read Api Keys y Write Api Key.

A continuación pasamos a describir en profundidad la funcionalidad del sistema End-to-End. Para ello hemos dividido su descripción en cuatro bloques. El primero es el entorno de ThingSpeak desde el cual el usuario tras registrarse creará un canal para seleccionar qué clase de sensores quiere utilizar. Además, de este entorno obtendremos las indicaciones necesarias para el uso correcto de la REST API que permita tanto a la aplicación móvil como al desarrollo web comunicarse con la base de datos de ThingSpeak donde se almacenan los datos de los sensores. El segundo es el entorno de compilación mediante la herramienta INTEL XDK donde abordamos el uso y las capacidades que nos proporciona la herramienta. El tercero es el entorno de la aplicación móvil desde donde el usuario gestiona los sensores de su vivienda. En este bloque también se explica con detalle las funcionalidades de la aplicación y la implementación de dichas funcionalidades. Y el cuarto bloque es el entorno del desarrollo web para emular la comunicación entre sensores, Raspberry y base de datos, desde donde se producen las peticiones HTTP para actualizar los valores de los sensores en la base de datos y también desde donde se visualizan los valores actualizados que se encuentran en la base de datos.

Por tanto, pasamos a describir el entorno de ThingSpeak.

1.4 Entorno de ThingSpeak

ThingSpeak es una plataforma abierta creada para el Internet de las Cosas (IoT: Internet of Things) que mediante una REST API permite interactuar de una forma sencilla con

una base de datos donde se almacenan y recuperan datos mediante el envío peticiones y respuestas HTTP a través de Internet.

Se ha utilizado ThingSpeak como entorno para almacenar los datos proporcionados por los sensores porque se trata de una plataforma abierta para el Internet de las Cosas que permite recopilar, almacenar, analizar, visualizar y actuar sobre la información recogida en sensores y dispositivos como aplicaciones web y móviles, redes sociales como Twitter, soluciones de mensajería, VoIP y nube como Twilio, hardware de código abierto como Arduino, Raspberry Pi o BeagleBone o con lenguajes de cálculo computacional como MATLAB. Además su uso es muy sencillo y la curva de aprendizaje es muy pequeña y rápida.

En este apartado se pasa a detallar tanto el entorno web proporcionado por ThingSpeak como las funcionalidades de la REST API proporcionada por ThingSpeak para la comunicación con su base de datos, permitiéndonos realizar los desarrollos de la aplicación móvil y del entorno web que emula a los sensores y la Raspberry. Comenzaremos por el entorno web.

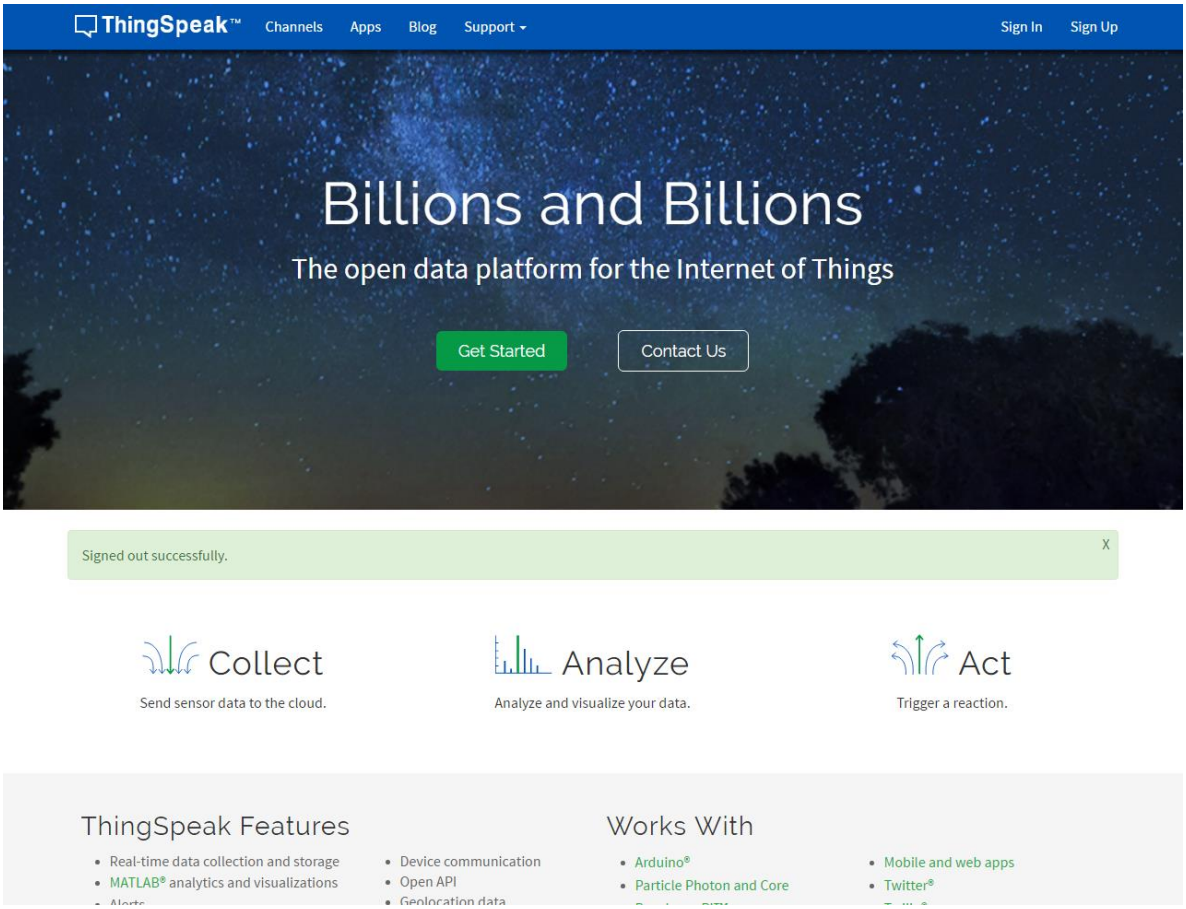

Figura 1.2. Acceso a www.thingspeak.com

Tras acceder al entorno web de ThingSpeak (www.thingspeak.com) nos encontramos con los siguientes elementos (Figura 1.2):

- Channels: Canales sobre la base de datos donde se almacenan los datos de los usuarios. Estos canales pueden ser públicos o privados.

- Apps: Servicios que ofrece ThingSpeak para tratar los datos almacenados. Algunos de estos servicios son mensajería push al entorno de Twitter, análisis de los datos mediate MATLAB o visualización de los datos almacenados mediante gráficas de MATLAB.

- Blog: Noticias de la comunidad de ThingSpeak.

- Support: Documentación, tutoriales y ejemplos de los usos de ThingSpeak.

- Sign In: Acceder mediante usuario y contraseña ya creados a un espacio propio en el entorno de ThingSpeak.

- Sign Up: Registrarse en la plataforma de ThingSpeak para poder hacer uso de sus servicios.

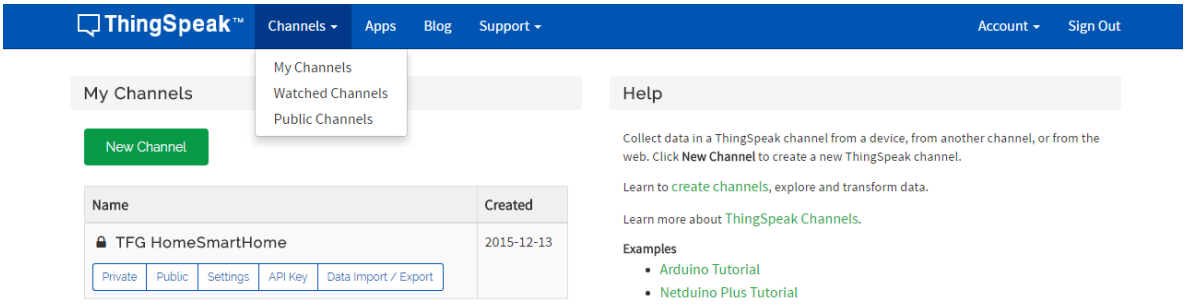

Figura 1.3. Creación de un nuevo canal

Una vez hemos completado el registro de un nuevo usuario al acceder desde la pestaña "Channels" a la pestaña "My Channels" (Figura 1.3). A través del botón "New Channel" se pueden crear canales desde donde configurar el nombre los sensores que se van a utilizar (Figura 1.4).

Los elementos que nos encontramos son los siguientes (Figura 1.4 y Figura 1.5):

- Name: Nombre para el canal creado.

- Description: Descripción para el canal creado.

- Fields: Nombre de los campos disponibles en el canal que hagan referencia a los sensores. Por cada canal se pueden habilitar hasta un máximo de ocho campos. Dado que los sensores por defecto que se han creado son interruptores de luz, reguladores de luz, sensores de intensidad luminosa, sensores de temperatura, sensores de humedad y sensores de lluvia los nombres disponibles que se pueden colocar sobre los ocho campos, siempre y cuando no se repita el mismo nombre en dos o más campos del mismo canal, son:

    - Interruptores de luz: light_s0, light_s1, light_s2, light_s3, light_s4, light_s5, light_s6 y light_s7.

    - Reguladores de luz: light_r0, light_r1, light_r2, light_r3, light_r4, light_r5, light_r6 y light_r7.

    - Sensores de intensidad luminosa: light_i0, light_i1, light_i2, light_i3, light_i4, light_i5, light_i6 y light_i7.

- Sensores de temperatura: temp0, temp1, temp2, temp3, temp4, temp5, temp6 y temp7.

- Sensores de humedad: hum0, hum1, hum2, hum3, hum4, hum5, hum6 y hum7.

- Sensores de lluvia: rain0, rain1, rain2, rain3, rain4, rain5, rain6 y rain7.

- Metadata: Campo para incluir información acerca del canal.

- Tags: Etiquetas para identificar el canal.

- Make Public: Hacer público el canal si se desea que lo pueda ver cualquier persona que acceda a la web.

- URL: Introducir la URL de una web que contenga información acerca del canal.

- Elevation: Especificar la posición de donde se encuentren los sensores.

- Show Location: Habilitar las opciones de “Latitude” y “Longitude”.

- Latitude: Especificar la posición de los sensores en grados decimales.

- Longitude: Especificar la posición de los sensores en grados decimales.

- Show Video: Habilitar “Video ID”.

- Video ID: Vídeo procedente de YouTube™ o Vimeo® que muestre información del canal.

- Show Status: Estado del canal.

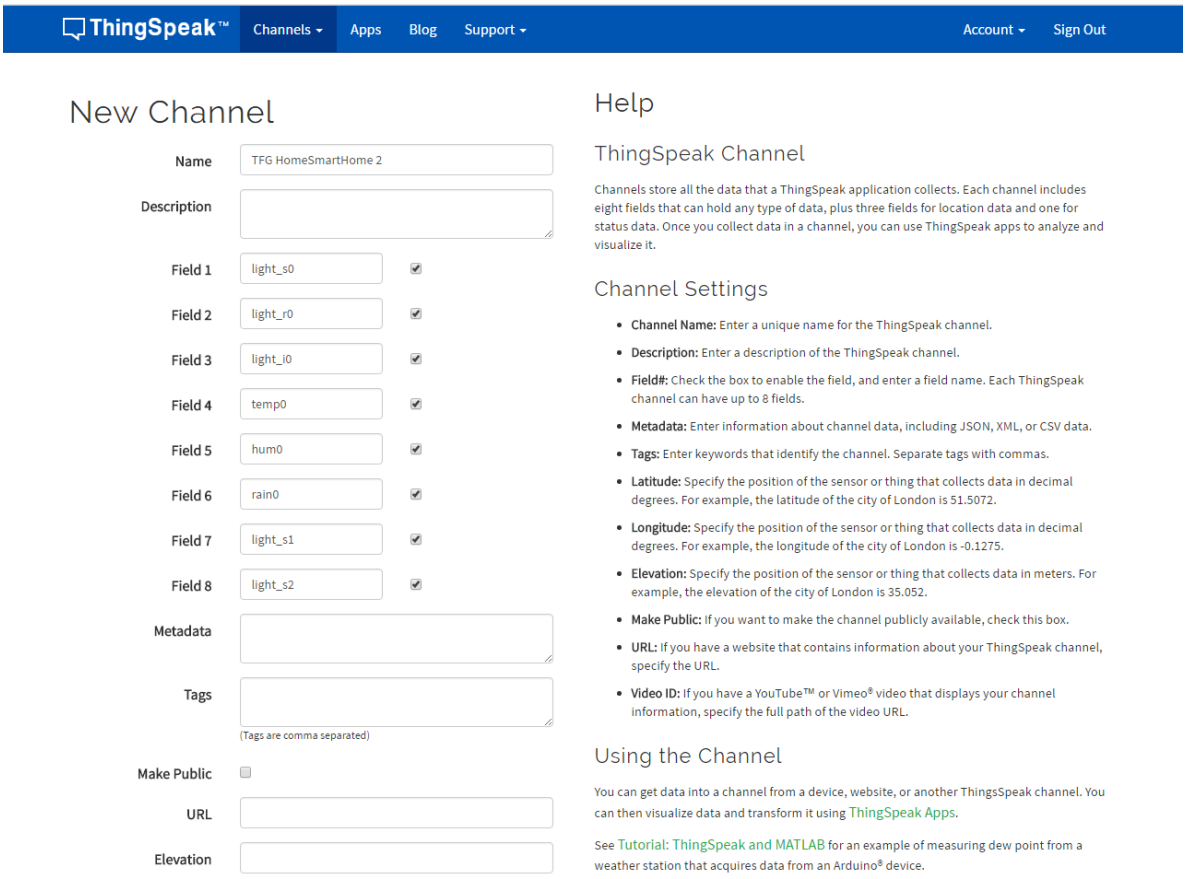

Figura 1.4. Datos de un nuevo canal, primera parte

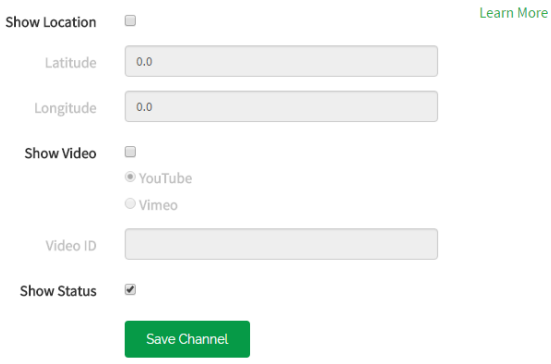

Figura 1.5. Datos de un nuevo canal, segunda parte

Tras crear el nuevo canal y acceder a dicho canal (Figura 1.6) al acceder a la pestaña "API Keys" podremos obtener los tres parámetros necesarios para introducir en la aplicación móvil una vez la hayamos instalado que se utilizarán en las peticiones HTTP para poder llevar a cabo la comunicación con la base de datos a través de la REST API. De esta forma la aplicación móvil se auto-configurará para mostrar los datos que se encuentren en la base de datos. Estos tres parámetros son:

- Channel ID: En el ejemplo 71882.

- Write API Key: En el ejemplo BBVUUT4GJO21D5C2.

- Read API Keys: En el ejemplo FYU7H1REB5VG7LI2.

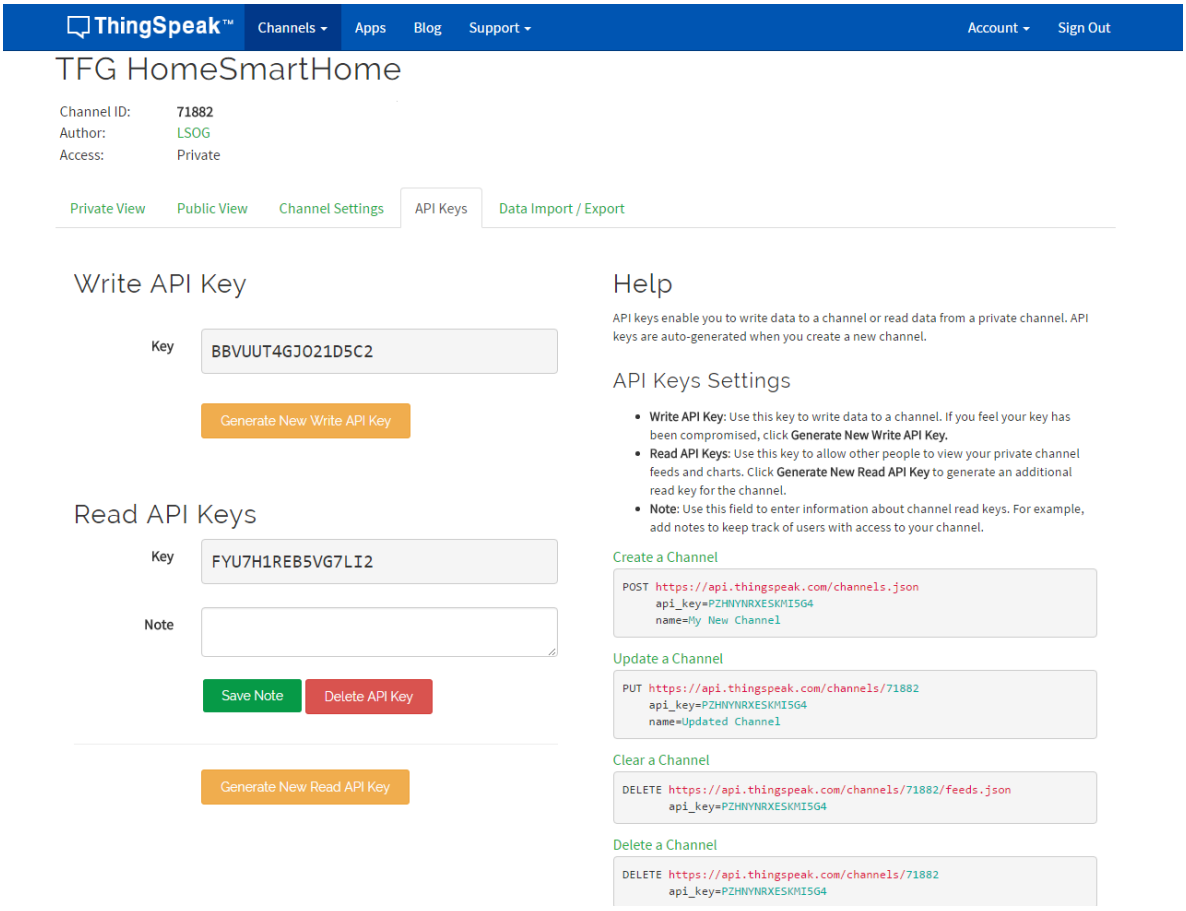

Figura 1.6. Parámetros Channel ID, Write API Key y Read API Keys

Ahora que ya conocemos los aspectos básicos del entorno web de ThingSpeak, es decir, la creación de una cuenta, la creación de un nuevo canal con sus respectivos campos a utilizar y la identificación de los tres parámetros necesarios para la auto-configuración del aplicativo móvil, pasamos a detallar el uso de la REST API proporcionada por ThingSpeak para poder actualizar el valor de los campos en la base de datos del canal creado mediante peticiones HTTP.

Para hacer uso de la REST API de ThingSpeak nos hemos basado en el manual de usuario proporcionado por el propio entorno (https://es.mathworks.com/help/thingspeak/channels-and-charts.html) donde se muestran las diversas peticiones HTTP que se deben realizar.

Para la obtención del último valor introducido en cualquiera de los ocho campos del canal creado utilizamos una petición HTTP con el método GET (Código 1.1) en la que introduciremos el parámetro Channel ID, el número del campo que queremos leer el cual variará desde el uno hasta el ocho y el parámetro Read API Keys.

```
https://api.thingspeak.com/channels/(Channel_ID)/fields/(Número_del_campo)/last.json?api_key=(Read_API_Keys)
```

Código 1.1. Petición HTTP con el método GET para lectura de último valor de uno de los ocho campos del canal

La respuesta que se recibe al realizar la petición HTTP con el método GET (Código 1.1) es un objeto JSON con el valor más actual del campo que hemos elegido en la petición HTTP (Código 1.2).

```
{
  "created_at":"2016-02-10T16:34:15Z",
  "entry_id":877778,
  "field4":"31.8"
}
```

Código 1.2. Ejemplo del objeto JSON recibido como respuesta a la petición HTTP realizada con el método GET para lectura del último valor de uno de los ocho campos del canal

Para la lectura del nombre que se ha introducido en cada uno de los ocho campos disponibles en el canal creado se realiza una petición HTTP con el método GET (Código 1.3) en la que introduciremos el parámetro Channel ID y el parámetro Read API Keys. Esta petición será utilizada por la aplicación móvil para interpretar qué campos se deben visualizar.

```
https://api.thingspeak.com/channels/(Channel_ID)/feeds.json?api_key=(Read_API_Keys)&results=1
```

Código 1.3. Petición HTTP con el método GET para lectura del nombre de cada uno de los ocho campos del canal

La respuesta que se recibe al realizar la petición HTTP con el método GET (Código 1.3) es un objeto JSON con el nombre de los campos activados en el canal creado de ThingSpeak (Código 1.4).

```
{
  "channel":
  {
    "id": 9,
```

```
    "name": "my_house",
    "description": "Netduino Plus connected to sensors around the house",
    "latitude": "40.44",
    "longitude": "-79.996",
    "field1": "Light",
    "field2": "Outside Temperature",
    "created_at": "2010-12-13T20:20:06-05:00",
    "updated_at": "2014-02-26T12:43:04-05:00",
    "last_entry_id": 6060625
  },
  "feeds":
  [
    {
      "created_at": "2014-02-26T12:42:49-05:00",
      "entry_id": 6060624,
      "field1": "188",
      "field2": "25.902335456475583"
    }
  ]
}
```


Código 1.4. Ejemplo del objeto JSON recibido como respuesta a la petición HTTP realizada con el método GET para lectura del nombre de los campos disponibles en el canal

Para la escritura nuevos valores sobre cualquiera de los ocho campos del canal creado utilizamos una petición HTTP con el método POST (Código 1.5) en la que introduciremos el parámetro Write API Key, el campo sobre el que queremos actualizar el valor donde podremos escribir desde field1 hasta field8 y el nuevo valor numérico que queremos almacenar en la base de datos.

```
https://api.thingspeak.com/update.json?api_key=(Write_API_Key)&(fieldX)=(Nuevo
_valor)
```

Código 1.5. Petición HTTP con el método POST para escritura de un nuevo valor en cualquiera de los ocho campos del canal

Acabamos de ver cómo utilizar el entorno web de ThingSpeak para obtener los tres parámetros necesarios (Channel ID, Write API Key y Read API Keys) para ser usados en la aplicación y cómo configurar los distintos campos que van a ser utilizados en función de los sensores de los que se disponga. Por otro lado, también hemos visto cuáles son las peticiones GET y POST básicas que utilizaremos para realizar lecturas y escrituras sobre la base de datos mediante peticiones HTTP que serán gestionadas por la REST API de

ThingSpeak y cuáles son las respuestas de objetos JSON que recibimos a realizar dichas peticiones HTTP. Con todo ello seremos capaces de realizar los desarrollos para la aplicación móvil y para el entorno web que simule los sensores y la Raspberry.

Ahora nos toca ver en qué consiste la herramienta INTEL XDK y cómo se utiliza.

1.5 Entorno de compilación mediante la herramienta INTEL XDK

INTEL XDK es una herramienta para desarrollar aplicaciones multiplataforma utilizando tecnologías estándar tales como HTML5, CSS3 y JAVASCRIPT. Con INTEL XDK, se pueden simular dispositivos con sistema operativo Android, con sistema operativo iOS y con el sistema operativo de Windows Phone. INTEL XDK nos permite desde una misma base de código generar aplicaciones para distintas plataformas tales como Android, iOS, Windows 10, Windows 8 y Windows Phone.

En este apartado se pasa a detallar el uso y las funcionalidades que aporta la herramienta de compilación, llamada INTEL XDK, utilizada para generar la aplicación para los smartphones.

Una vez hayamos descargado en instalado en nuestro ordenador la aplicación de INTEL XDK (https://software.intel.com/es-es/intel-xdk) para comenzar a hacer uso de la herramienta lo primero que debemos hacer tras registrarnos es la creación de un nuevo proyecto (Figura 1.7).

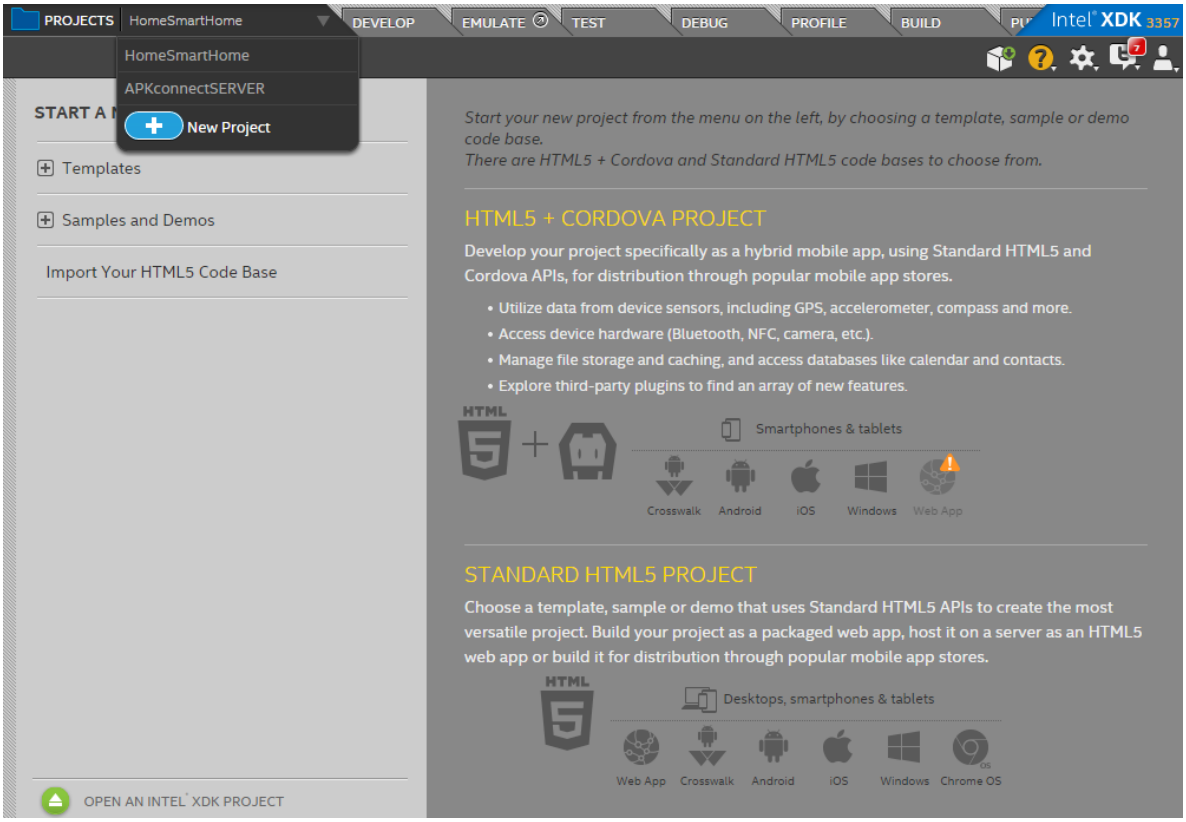

Figura 1.7. Creación de un nuevo proyecto en INTEL XDK

Una vez hemos creado el nuevo proyecto debemos seleccionar el tipo de plantilla a utilizar. En nuestro caso una plantilla estándar HTML5 en blanco (Figura 1.8).

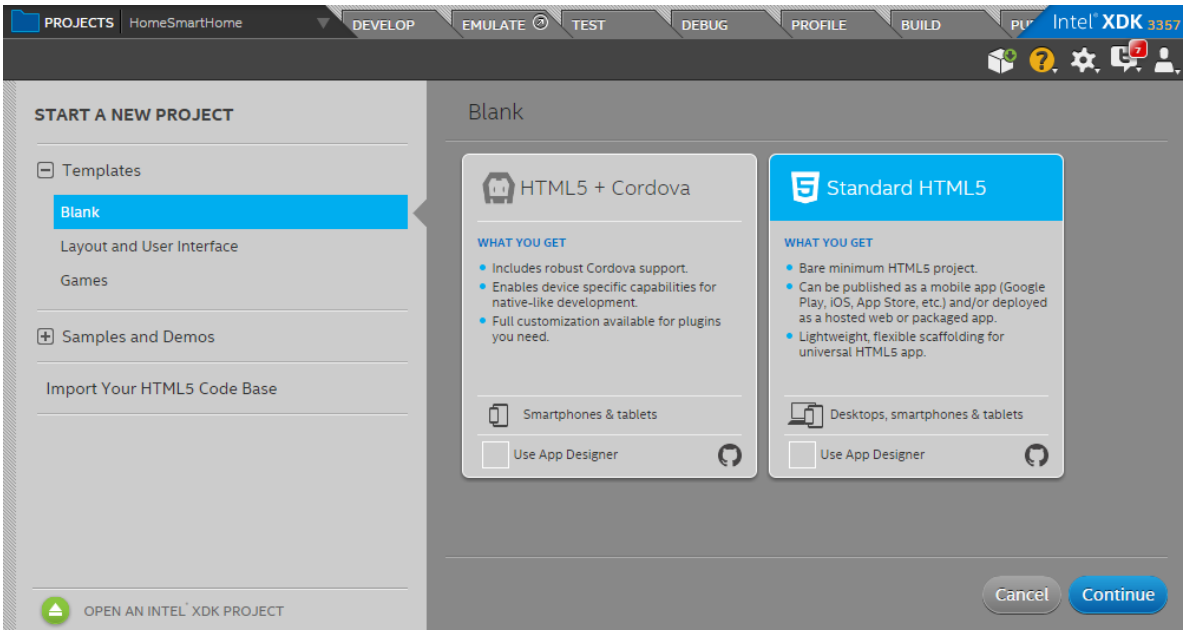

Figura 1.8. Selección de nueva plantilla en INTEL XDK

Ahora debemos elegir el nombre que queremos que tenga nuestra aplicación, en nuestro caso es "Home Smart Home" (Figura 1.9).

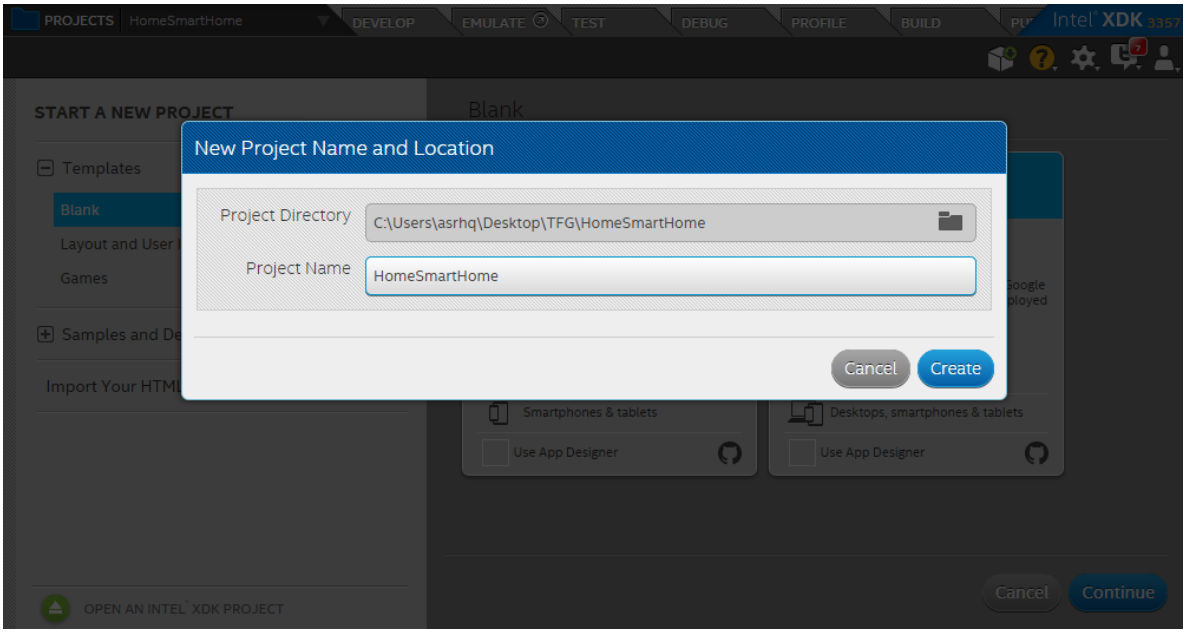

Figura 1.9. Nombre de nuestra aplicación en INTEL XDK

Una vez que ya hemos creado el proyecto es momento de configurar las opciones de compilación para, en nuestro caso, generar la aplicación para dispositivos Android. Para ello, desde la esquina superior izquierda accederemos a "PROJECTS" y seleccionaremos nuestro proyecto actual. Iremos al apartado "HYBRID MOBILE APP BUILD SETTINGS" y seleccionaremos únicamente el icono de Android (Figura 1.10).

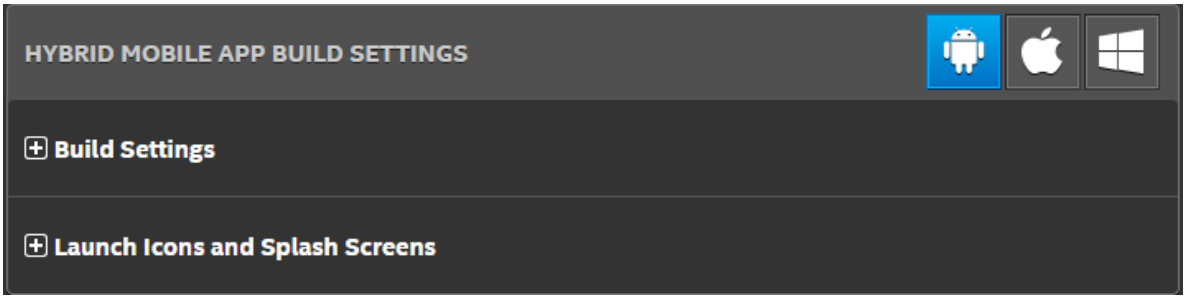

Figura 1.10. Selección de Android para configuración de su entorno de compilación

Ahora procedemos a desplegar la pestaña "Build Settings" para realizar los ajustes necesarios para la correcta compilación del programa (Figura 1.11 y Figura 1.12) podremos configurar:

- Optimize with Crosswalk: utilizado para optimizar el comportamiento de la aplicación compilada en los dispositivos Android. Lo dejamos desmarcado para un comportamiento estándar de la aplicación

- App ID: Identificador de la aplicación. Al no marcar optimización con crosswalk este ID debe ser xdk.intel.blank.template.

- App Name: Nombre que queremos que tenga la aplicación.

- App Description: Descripción de la aplicación. Descripción deseada.

- Author: Autor de la aplicación.

- App Version: Versión de la aplicación. En nuestro caso se trata de la versión 4.2

- Cordova CLI Version: Versión de Cordova que se utiliza para la compilación. En nuestro caso trabajaremos con la versión 5.1.1.

- Whitelist: Define los recursos internos y externos a los que la aplicación tiene el acceso permitido. En nuestro caso trabajaremos con "Cordova Legacy Whitelist", en "Network Request" utilizaremos un asterisco (*) para indicar que queremos permitir todas las solicitudes de acceso a la red e "Intent" lo dejaremos en blanco para bloquear todas las solicitudes de inteción.

- Fullscreen: Deshabilitaremos la opción de pantalla completa para poder utilizar en nuestros dispositivos Android los botones de atrás y salir al menú principal.

- Developer Certificate: Generaremos un nuevo certificado para poder firmar la aplicación. Para ello seleccionamos la creación de un nuevo certificado de desarrollador (Figura 1.13) y a continuación rellenamos los datos requeridos para posteriormente salvarlos (Figura 1.14).

- Orientation: La orientación de la aplicación puede ser en vertical o en horizontal. En nuestro caso la dejamos como "default" lo que implica que tendrá orientación vertical.

- App Version Code: Versión del código de la aplicación. Lo normal es aumentar este número en uno cada vez que se genera una nueva versión.

- Minimum Android API: Mínimo nivel de la API de Android en la cual la aplicación puede ser instalada y puede funcionar correctamente. En nuestro caso la catorce.

- Target Android API: Nivel de la API de Android contra la que la aplicación ha sido probada. En nuestro caso veintiuno.

- Install Location: Determina donde será instalada la aplicación en el dispositivo si en la memoria interna o en la memoria externa que pueda tener el dispositivo. En nuestro caso seleccionamos que lo instale de forma automática donde crea conveniente.

- Signed: Lo marcamos para firmar la aplicación al compilarla.

- Add Permisions: Añadir permisos a las opciones de compilación. En nuestro caso no queremos añadir ningún permiso adicional por lo que lo dejamos en blanco.

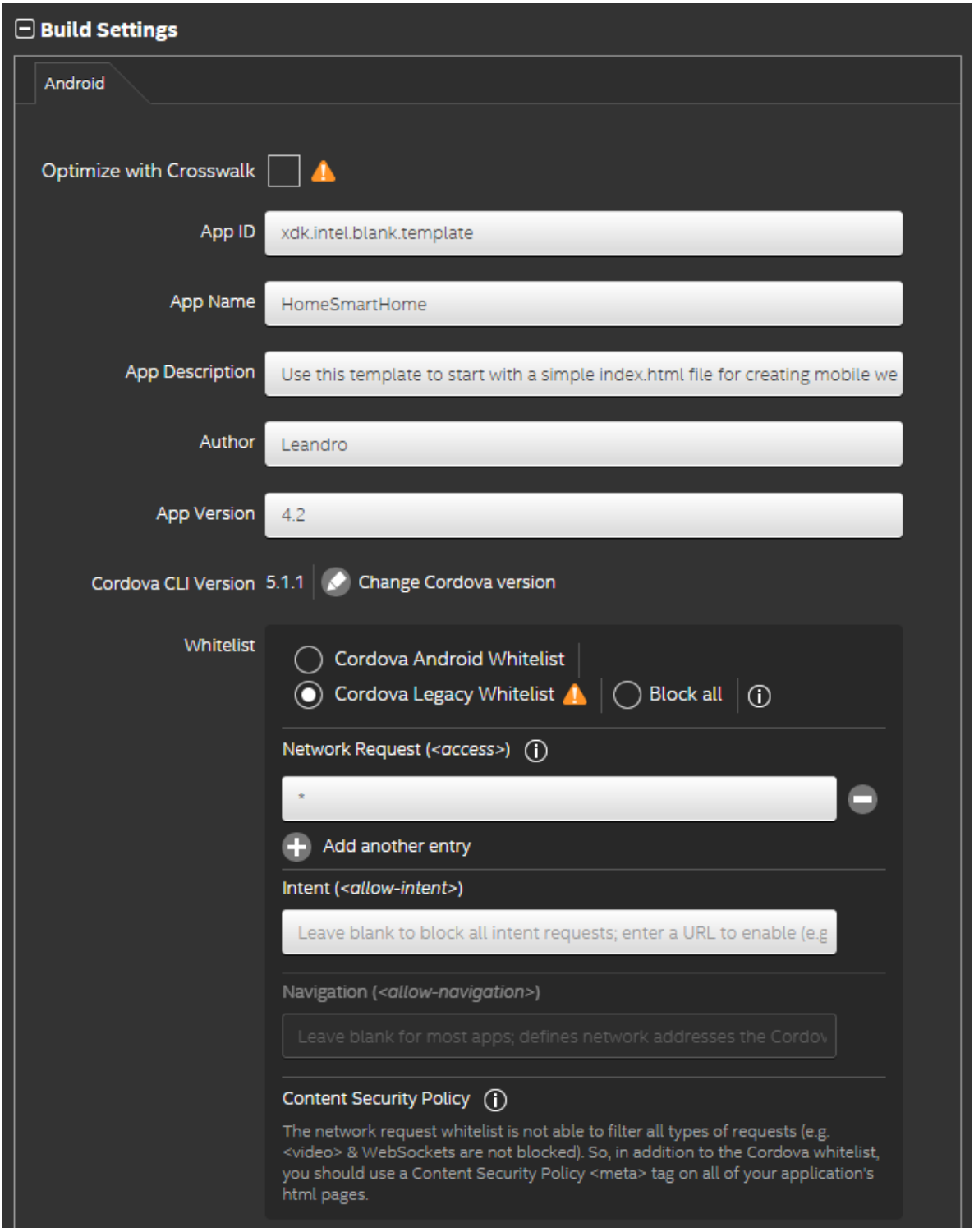

Figura 1.11. Configuración de las opciones de compilación, primera parte

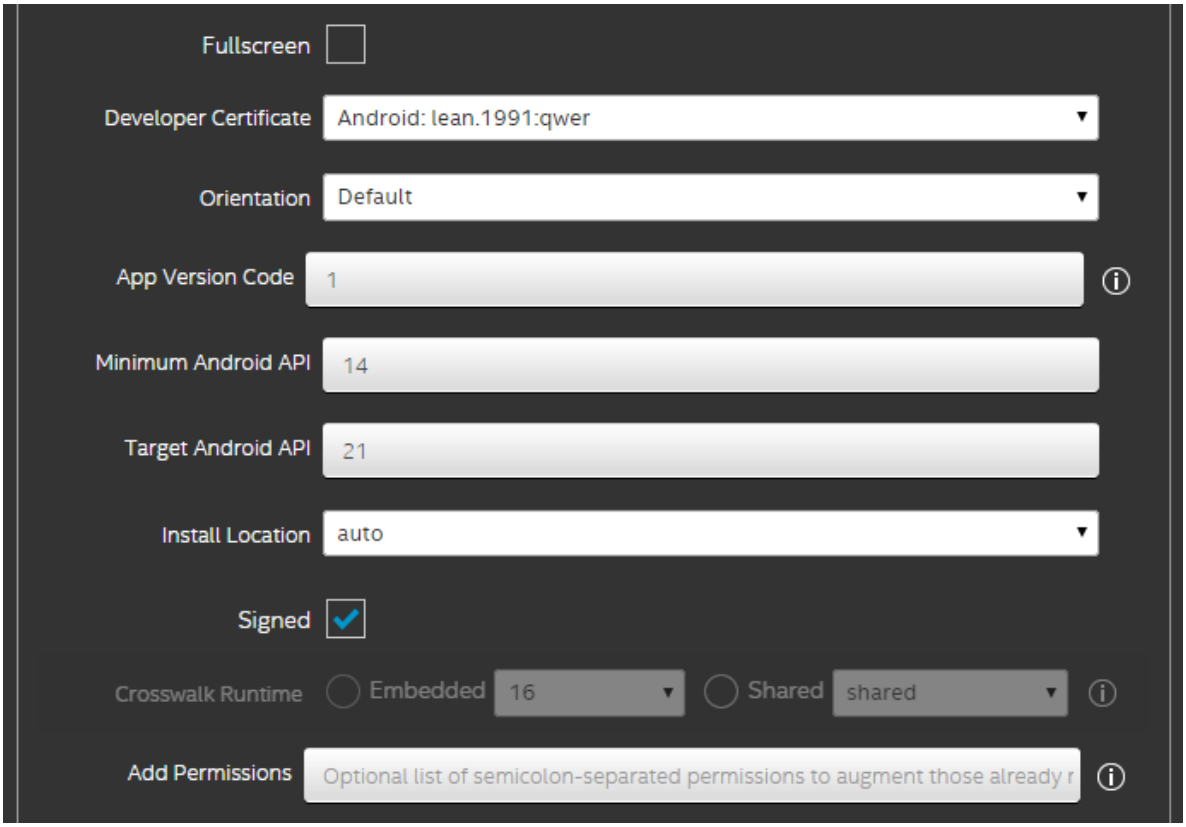

Figura 1.12. Configuración de las opciones de compilación, segunda parte

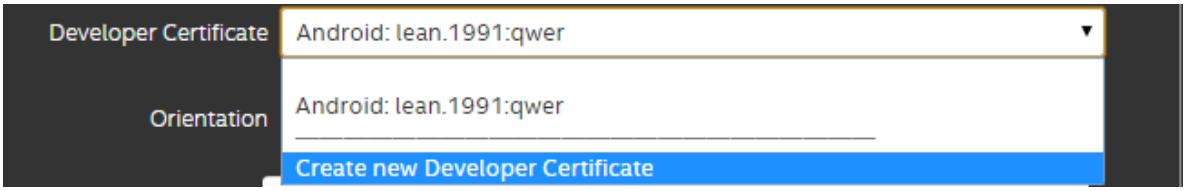

Figura 1.13. Creación de un nuevo certificado de desarrollador para firmar las aplicaciones

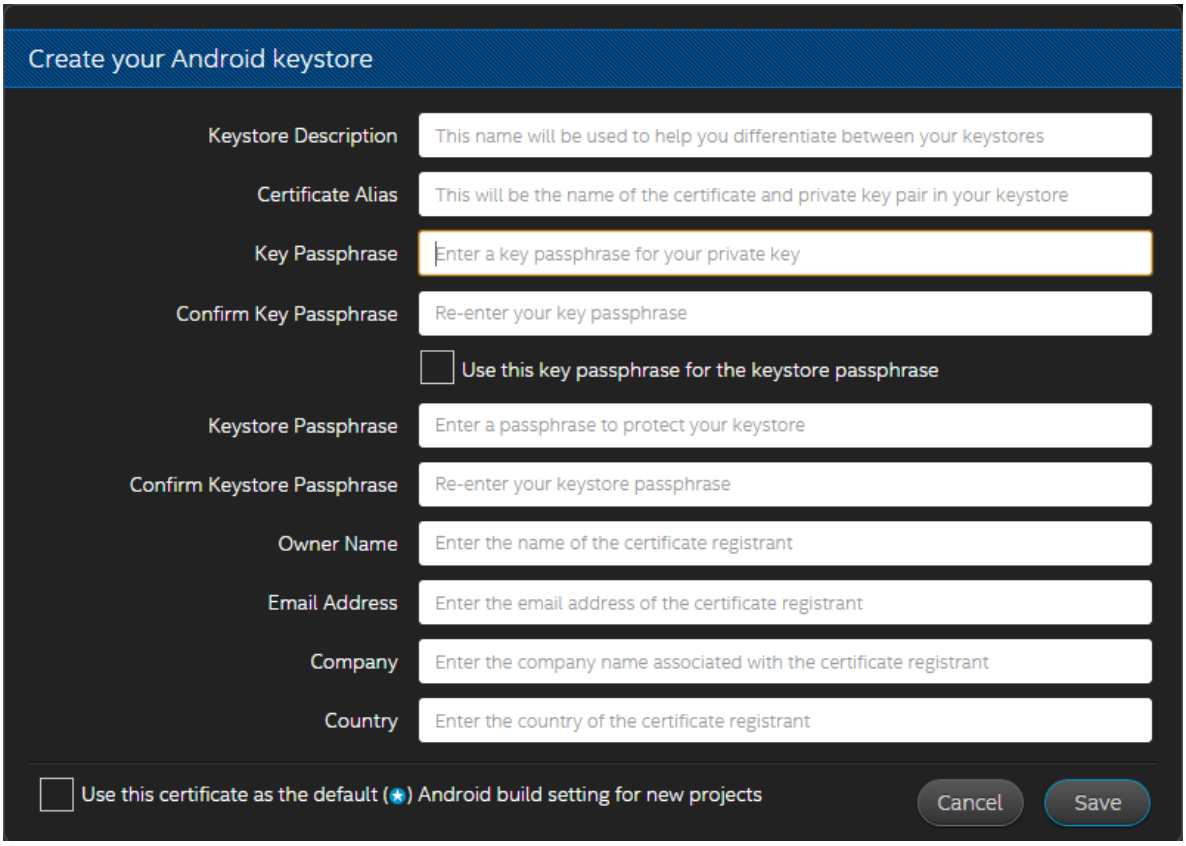

Figura 1.14. Datos a completar para la creación del certificado de desarrollador

Una vez configurados los parámetros de compilación vamos incluir los iconos que se mostrarán para iniciar la aplicación (Figura 1.15). Para ello desplegaremos la pestaña "Launch Icons and Spash Screens" (Figura 1.10) donde seleccionaremos en el apartado "Launch Icons" (Figura 1.16) las imágenes que vamos a utilizar. El apartado "Splash Screen" lo dejaremos vacío. Dichas imágenes deben estar previamente creadas y guardadas en el directorio donde hemos creado el proyecto (Figura 1.17). Además estas imágenes deben tener el formato PNG y sus medidas deben ser 36x36 píxeles, 48x48 píxeles, 72x72 píxeles y 96x96 píxeles.

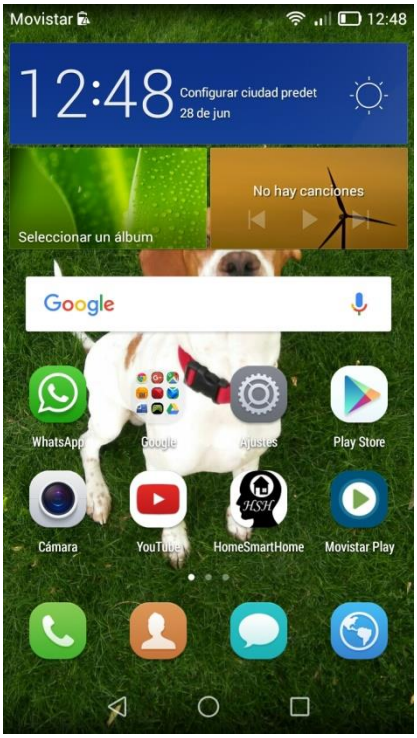

Figura 1.15. Icono de la aplicación móvil "Home Smart Home"

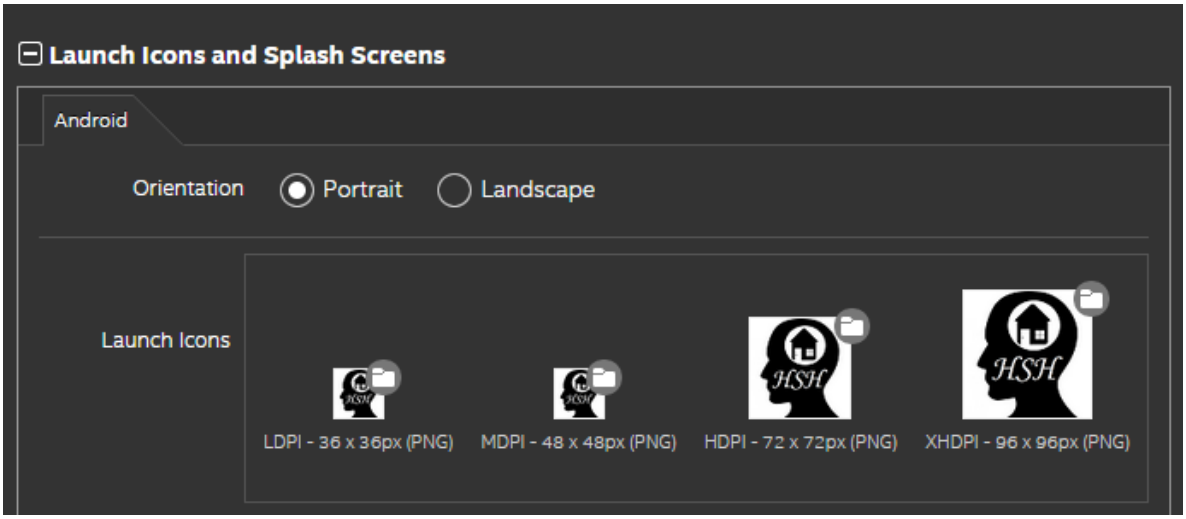

Figura 1.16. Selección de imágenes "Launch Icons"

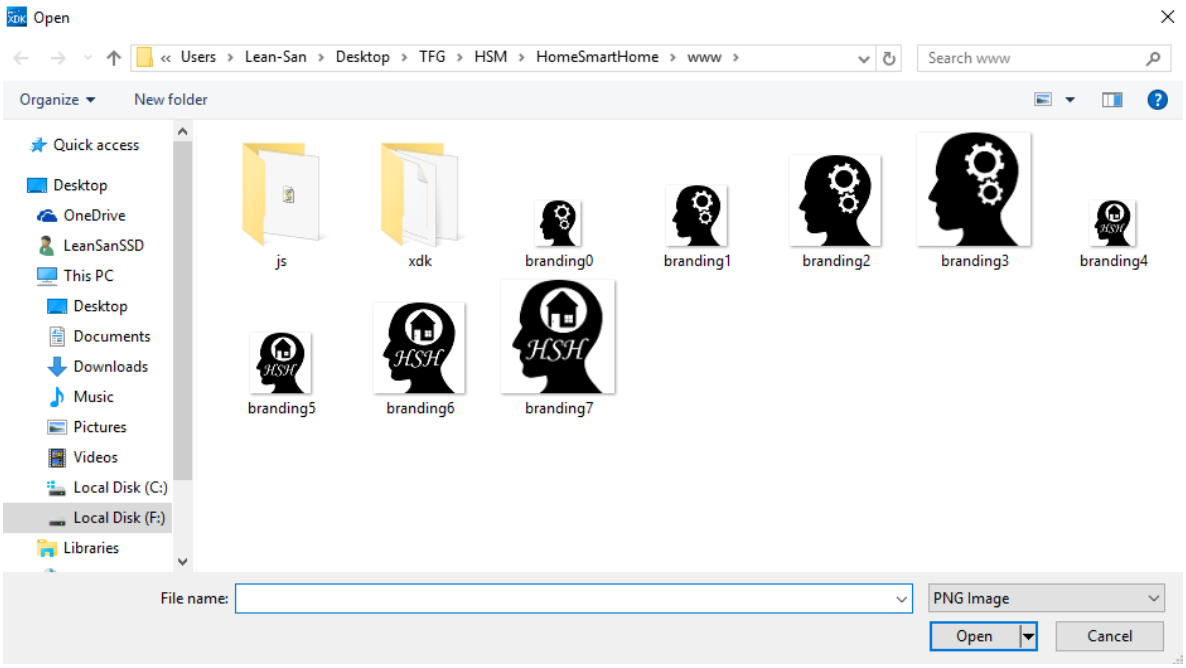

Figura 1.17. Almacenar imágenes en el directorio donde se ha creado el proyecto

Ya hemos finalizado la configuración de los parámetros de compilación y la creación e inclusión de los iconos que se mostrarán cuando se instale la aplicación y se desee arrancar. Por tanto procedemos a detallar dónde desarrollar el código de nuestra aplicación, dónde emular nuestra aplicación para diferentes dispositivos y dónde realizar la compilación de la aplicación para su posterior descarga.

Para poder empezar a desarrollar nuestro código iremos a la pestaña "DEVELOP" habiendo seleccionado previamente nuestro proyecto "HomeSmartHome" (Figura 1.18). En nuestro caso dentro de del archivo de trabajo index.html hemos desarrollado todo el código de la aplicación, es decir, tanto el código HTML5, como el CSS3 para los estilos, como finalmente el código JAVASCRIPT acompañado de librerías JQUERY para dotar de funcionalidad a la aplicación.

Una vez hemos finalizado de escribir el código de nuestra aplicación o, en su defecto, parte de dicho código procedemos a guardarlo desde la pestaña "File".

Tras haber guardado nuestro desarrollo lo que pretendemos a continuación es emular nuestra aplicación en distintos tipos de terminales para comprobar que su comportamiento es el esperado. Para ello accederemos a la pestaña "EMULATE" en donde dispondremos de una gran variedad de opciones de dispositivos móviles y tablets donde emular nuestra aplicación (Figura 1.19).

Una vez hemos finalizado el proceso de desarrollo y emulación de la aplicación y hemos comprobado que todo funciona correctamente es momento de comenzar el proceso de compilación de la aplicación para generar el fichero HSH_TFG.apk que permitirá instalar la aplicación en los dispositivos móviles.

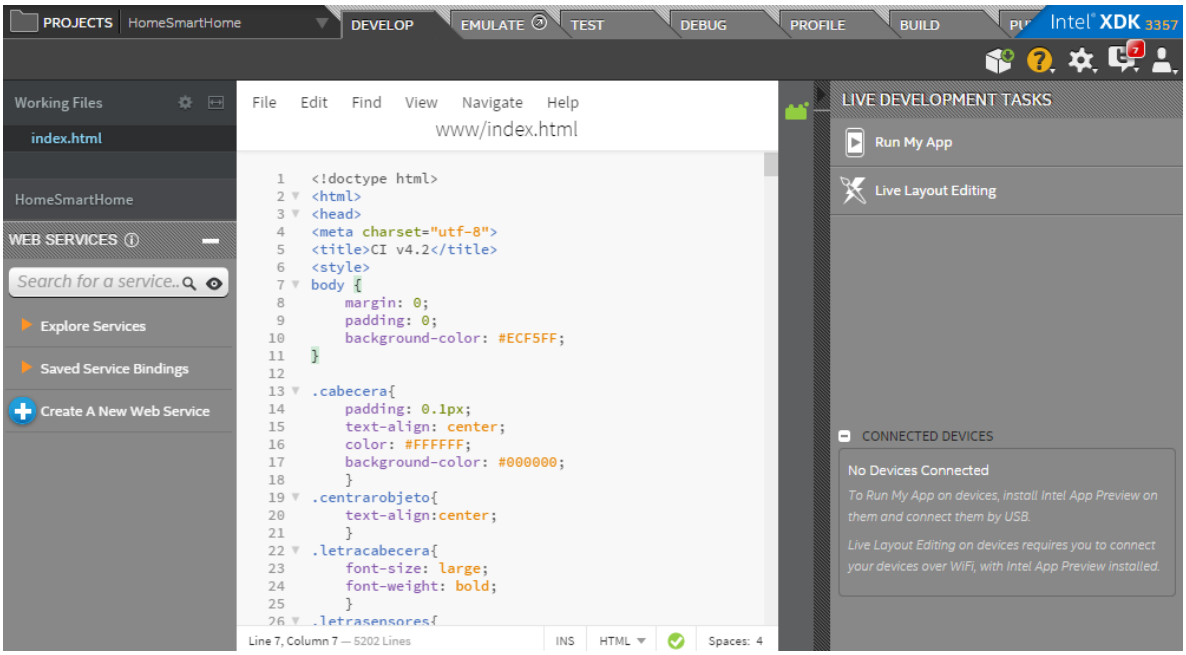

Figura 1.18. Pestaña "DEVELOP" en INTEL XDK

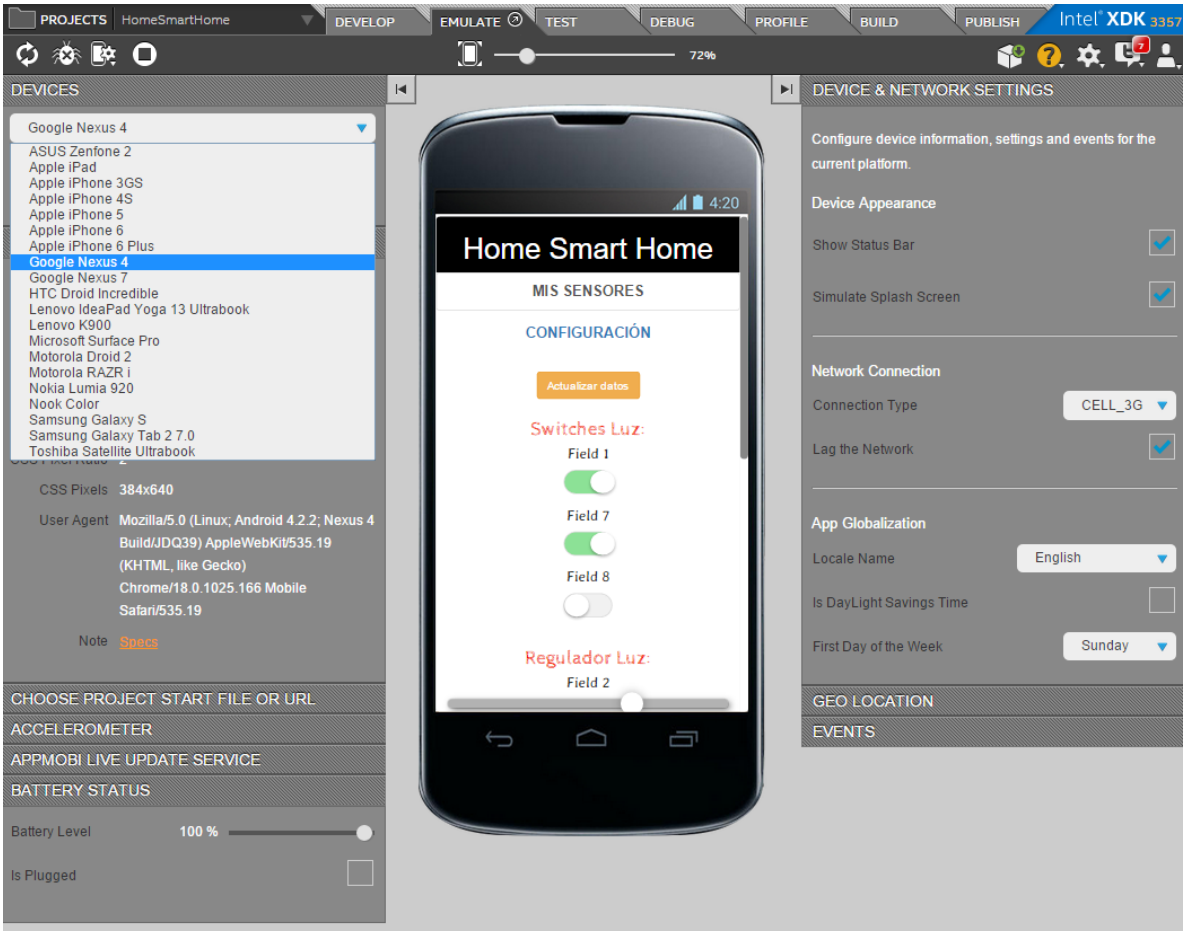

Figura 1.19. Pestaña "EMULATE" en INTEL XDK donde emular una gran variedad de smartphones y tablets

Accedemos a la pestaña "BUILD" para comenzar la compilación de nuestra aplicación (Figura 1.20).

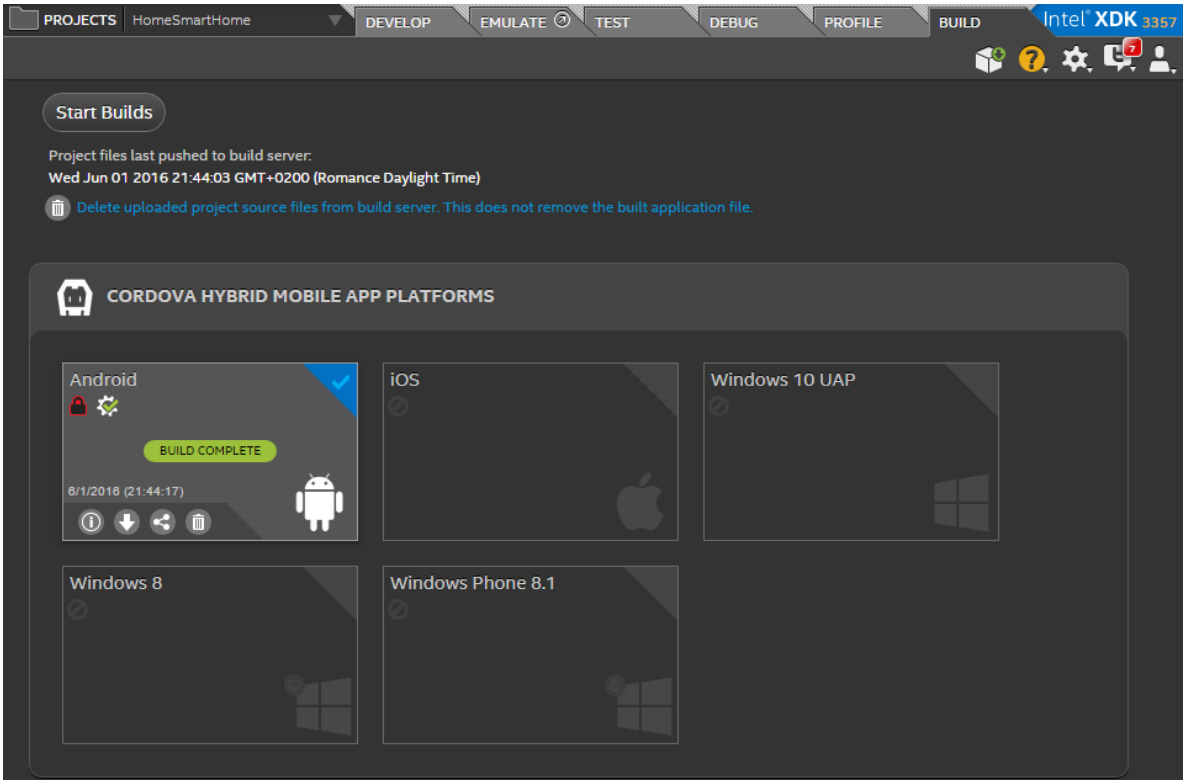

Figura 1.20. Pestaña "BUILD" para compilar la aplicación desarrollada

Dentro de "CORDOVA HYBRID MOBILE APP PLATFORMS" en Android seleccionamos el candado de color rojo para desbloquear nuestro certificado de desarrollador con el que podremos generar y firmar la nueva aplicación al mostrar el candado abierto de color negro (Figura 1.21 y Figura 1.22).

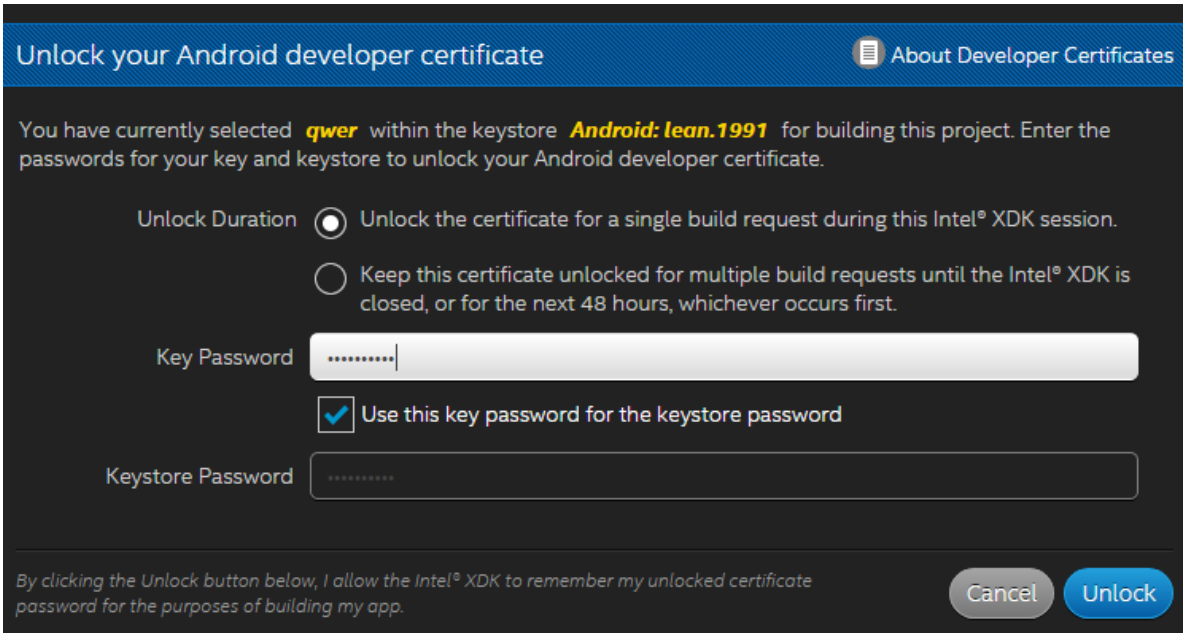

Figura 1.21. Desbloqueo de certificado de desarrollador

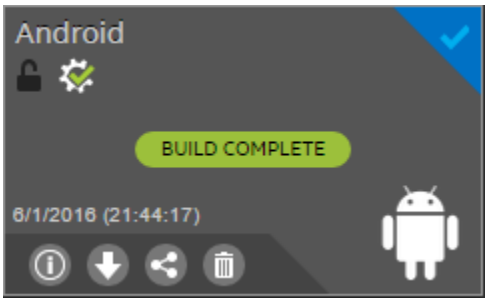

Figura 1.22. Certificado de desarrollador desbloqueado y compilación de la aplicación finalizada

Lo único que queda por hacer es pulsar el botón "Start Builds" (Figura 1.20) y esperar a que el proceso de generación de la aplicación haya terminado (Figura 1.22).

Tras haber generado la aplicación ya sólo nos queda poder descargarla para instalarla en nuestro dispositivo móvil, para ello pulsamos sobre el botón de descarga de la aplicación (Figura 1.23) y lo guardamos en nuestro ordenador para su posterior subida al market de Google Play o a un repositorio desde donde poder descargarlo desde un Smartphone Android para su instalación y uso.

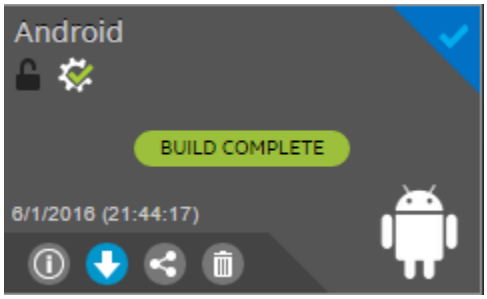

Figura 1.23. En color azul queda marcado el botón de descarga del aplicativo generado

Cabe mencionar que desde la herramienta INTEL XDK también se pueden publicar directamente en diferentes markets la aplicación generada. Esto se lleva a cabo desde la pestaña "PUBLISH" de la herramienta (Figura 1.24).

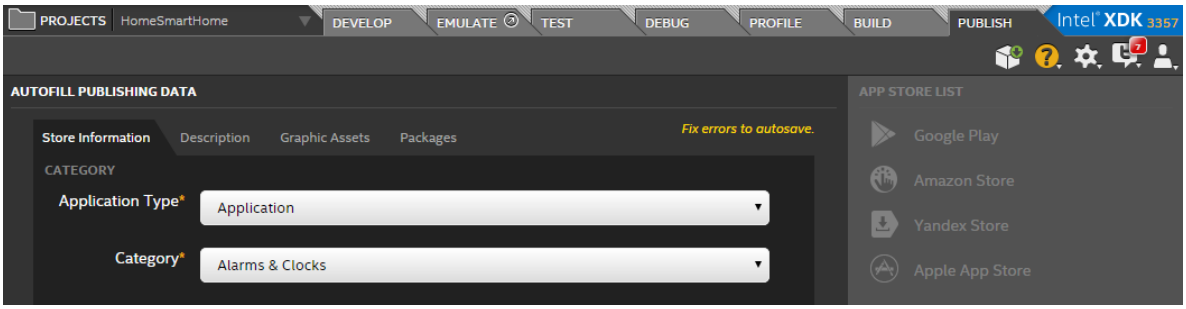

Figura 1.24. Pestaña "PUBLISH" desde donde se puede publicar directamente la aplicación generada en distintos markets

Acabamos de ver cómo utilizar la herramienta INTEL XDK y varias de las funcionalidades que nos ofrece para desarrollar una aplicación que pueda ser emulada antes de poder compilarla para diversas plataformas como pueden ser Android, iOS, Windows Phone, Windows 8 y Windows 10. Aunque en nuestro caso únicamente hemos generado el aplicativo para dispositivos Android.

Ahora que ya conocemos los conceptos básicos del uso de ThingSpeak y sabemos cómo manejar la herramienta INTEL XDK para generar nuestro aplicativo pasamos a ver el entorno de la aplicación móvil donde se describe tanto el entorno gráfico de la aplicación y sus funcionalidades como la implementación de dichas funcionalidades.

1.6 Entorno de la aplicación móvil

En este apartado se pasa a detallar el entorno gráfico desarrollado en la aplicación mientras que al mismo tiempo se pretende explicar las funcionalidades de este entorno gráfico y cómo se han implementado dichas funcionalidades. Para ello hemos divido esta

explicación el dos bloques, en el primero tratamos el entorno gráfico de la configuración de la aplicación mediante los tres parámetros de configuración llamados Channel ID, Write API Key y Read API Keys, además de la opción de personalización del nombre de los campos para poder identificar nuestros sensores en la aplicación de una forma sencilla y la implementación de todas estas funcionalidades. En el segundo bloque se detalla el entorno gráfico de los sensores, la opción de refresco de los datos que se muestran en la aplicación y la implementación de todas estas funcionalidades.

Antes de comenzar la explicación de los bloques de configuración y de identificación de sensores se aborda a modo de pseudocódigo la distribución establecida a la hora del desarrollo de la aplicación (Código 1.6). Contamos con las librerías de JQUERY añadidas por medio de un script, con las librerías de BOOTSTRAP para dotar de elegancia a la interfaz gráfica de la aplicación añadidas por medio de un script y de un link a una hoja de estilos y también para dotar de un estilo de tipografía diferente se han añadido por medio de otro script tipografías procedentes de edgefonts. Finalmente acerca del pseudocódigo de la aplicación existe un script más que es el que contiene toda la lógica de lectura de los campos del canal requerido de ThingSpeak, la escritura sobre dichos campos y la modificación en tiempo del código HTML5 de la interfaz gráfica de la apliación. Para este último script se ha utilizado JAVASCRIPT haciendo uso de las librerías de JQUERY. En lo referente al apartado donde se encuentra el código HTML5 que interpreta la interfaz gráfica de la aplicación lo dividimos en tres partes: la cabecera, los campos de los sensores y los campos de configuración de la aplicación donde introducir los parámetros Channel ID, Write API Key y Read API Keys, además de la configuración del nombre de los campos de los sensores dentro de la aplicación.

```
<!doctype html>
<html>
<head>
      <meta charset="utf-8">

      <title>CI v4.2</title>

      <style>
                  //CSS3 para mejorar gráficamente la aplicación
      </style>

      <meta name="viewport" content="width=device-width, initial-
      scale=1, user-scalable=no">

      <script
      src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.mi
      n.js"></script>
```

```
        <link rel="stylesheet"
        href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
        .min.css">

        <script
        src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.m
        in.js"></script>

        <script src="http://use.edgefonts.net/cabin-
        sketch:n4:default;wallpoet:n4:default;averia-gruesa-
        libre:n4:default.js"></script>

        <script>
              //FUNCIONALIDADES de lectura de campos de la base de datos,
              actualización de valores de los campos sobre la base de
              datos y modificación de código HTML5 en tiempo real sobre la
              aplicación para mostrar resultados. Se utilizan JAVASCRIPT y
              librerías de JQUERY.
        </script>
</head>

<body>
        <div>
              //CÓDIGO HTML5 que muestra la interfaz gráfica de la
              aplicación. Se divide en tres partes: la cabecera, los
              campos de los sensores y los campos de configuración de la
              aplicación donde introducir los parámetros Channel ID, Write
              API Key y Read API Keys, además de la configuración del
              nombre de los campos de los sensores dentro de la
              aplicación.
        </div> <!--div global-->
</body>
</html>
```

Código 1.6. Pseudocódigo de la aplicación "Home Smart Home"

Comenzamos el primer bloque detallando el entorno gráfico de la zona de configuración dentro de la aplicación móvil "Home Smart Home" y sus funcionalidades (Figura 1.25, Figura 1.26 y Figura 1.27). Dentro del campo de configuración encontramos las siguientes opciones:

- Introduce tu Channel ID: dentro de este campo debemos introducir el parámetro Channel ID obtenido en la plataforma de ThingSpeak cuando creamos nuestro canal (Figura 1.6).

- Introduce el Write API Key de tu channel: dentro de este campo debemos introducir el parámetro Write API Key obtenido en la plataforma de ThingSpeak cuando creamos nuestro canal (Figura 1.6).

- Introduce el Read API Keys de tu channel: dentro de este campo debemos introducir el parámetro Read API Keys obtenido en la plataforma de ThingSpeak cuando creamos nuestro canal (Figura 1.6).

- Guardar datos: En relación al Channel ID, Write API Key y Read API Keys (Figura 1.26). Guarda los datos en memoria local para que sean cargados una vez inicie la aplicación y de esta forma se puedan ejecutar las peticiones HTTP. Tras pulsar el botón de guardado se muestran los resultados almacenados en memoria (Figura 1.28, Figura 1.29 y Figura 1.30). En el caso de guardar valores en blanco lo que se mostrará son mensajes Alert con el contenido vacío para indicar que lo que se ha guardado son espacios en blanco.

- Mostrar datos: En relación al Channel ID, Write API Key y Read API Keys (Figura 1.26). Muestra los datos del Channel ID, Write API Key y Read API Keys guardados en memoria local. Tras pulsar el botón de mostrar datos hay dos opciones, en la primera si no hay ningún contenido guardado se mostrará un mensaje de Alert con el contenido "Sin datos" para indicar que no se ha introducido ningún dato (Figura 1.31) y en la segunda opción si se han guardado datos en memoria local se mostrará el contenido guardado (Figura 1.28, Figura 1.29, Figura 1.30).

- Borrar datos: En relación al Channel ID, Write API Key y Read API Keys (Figura 1.26). Borra los datos del Channel ID, Write API Key y Read API Keys guardados en memoria local. Tras pulsar el botón de borrado de datos se mostrarán tres mensajes de Alert con los contenidos "Introduce Channel ID", "Introduce WAK" e "Introduce RAK" donde WAK hace referencia a Write API Key y RAK a Read API Key (Figura 1.32, Figura 1.33 y Figura 1.34).

- Nombre field 1: Campo desde el que se puede modificar el nombre del campo uno del canal dentro de la aplicación (Figura 1.27).

- Nombre field 2: Campo desde el que se puede modificar el nombre del campo dos del canal dentro de la aplicación.

- Nombre field 3: Campo desde el que se puede modificar el nombre del campo tres del canal dentro de la aplicación.

- Nombre field 4: Campo desde el que se puede modificar el nombre del campo cuatro del canal dentro de la aplicación.

- Nombre field 5: Campo desde el que se puede modificar el nombre del campo cinco del canal dentro de la aplicación.

- Nombre field 6: Campo desde el que se puede modificar el nombre del campo seis del canal dentro de la aplicación.

- Nombre field 7: Campo desde el que se puede modificar el nombre del campo siete del canal dentro de la aplicación.

- Nombre field 8: Campo desde el que se puede modificar el nombre del campo ocho del canal dentro de la aplicación.

- Guardar datos: En relación en los campos que van desde field 1 hasta field 8. Guarda los datos en memoria local para que sean cargados una vez inicie la aplicación. Muestra los datos del campo seleccionado (desde el field 1 hasta el field 8) memoria local (Figura 1.35)

- Mostrar datos: En relación en los campos que van desde field 1 hasta field 8. Tras pulsar el botón de mostrar datos en los campos field 1 a field 8 se muestra en un Alert el resultado guardado o si no hay datos guardados muestra de "Field 1", "Field 2", "Field 3", "Field 4", "Field 5", "Field 6", "Field 7" o "Field 8" en función del campo en el que nos encontremos (Figura 1.36)

- Valor predeterminado: En relación en los campos que van desde field 1 hasta field 8. Borra los datos del campo seleccionado, desde field1 hasta field 8, guardados en memoria local. Tras pulsar el botón de valor predeterminado de los campos field 1 a field 8 se muestra en un Alert el mensaje "Valor por defecto restablecido" (Figura 1.37).

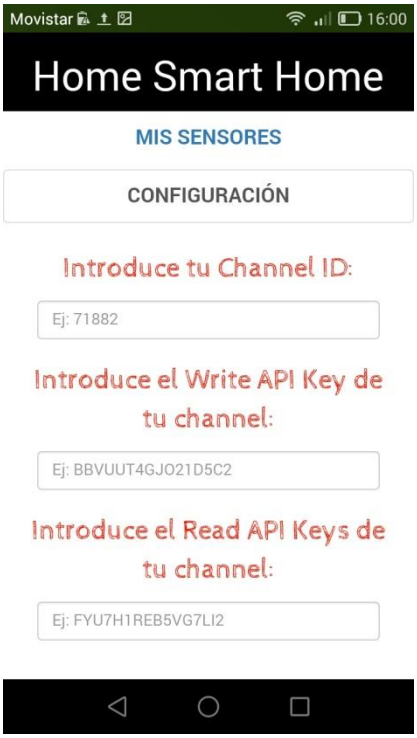

Figura 1.25. Zona configuración: Channel ID, Write API Key, Read API Keys

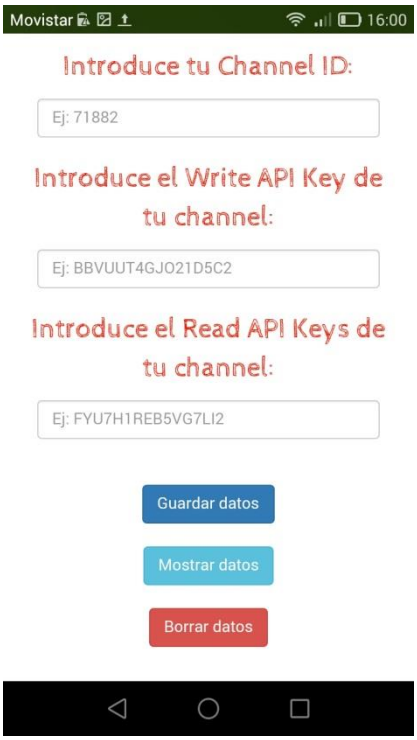

Figura 1.26. Zona de configuración: Botones de guardar datos, mostrar datos y borrar datos

Figura 1.27. Zona de configuración: Guardar, mostrar o restablecer al valor por defecto el nombre los campos de los sensores

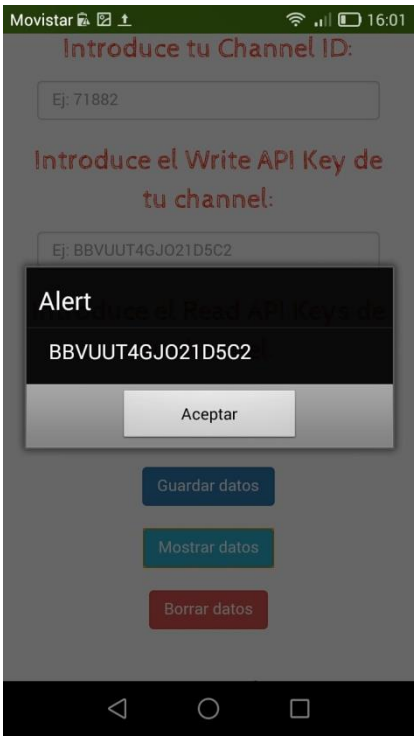

Figura 1.28. Tras pulsar el botón guardar datos o mostrar datos si han sido guardados se muestran los datos del Channel ID

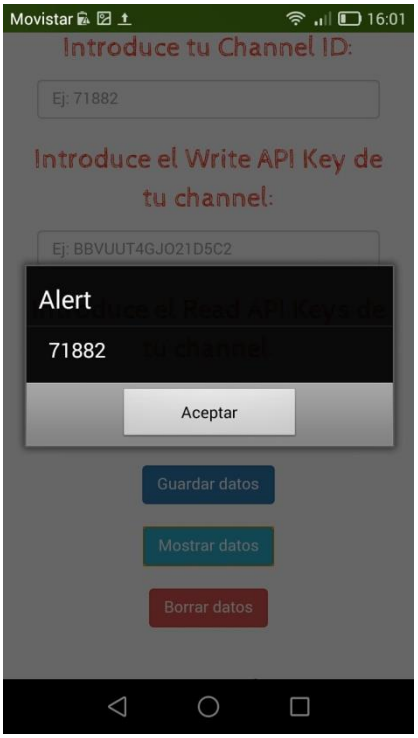

Figura 1.29. Tras pulsar el botón guardar datos o mostrar datos si han sido guardados se muestran los datos del Write API Key

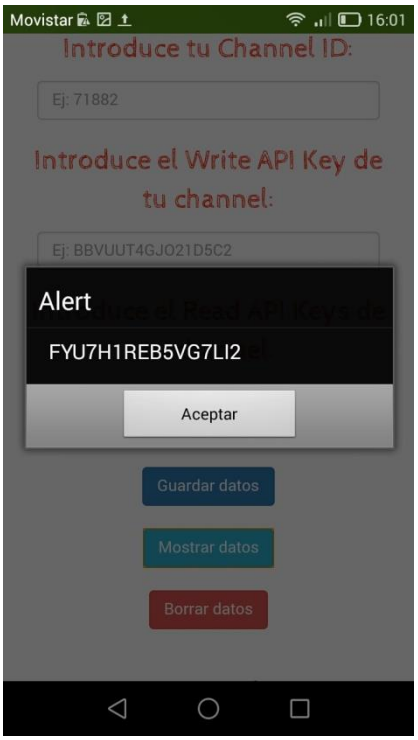

Figura 1.30. Tras pulsar el botón guardar datos o mostrar datos si han sido guardados se muestran los datos del Read API Keys

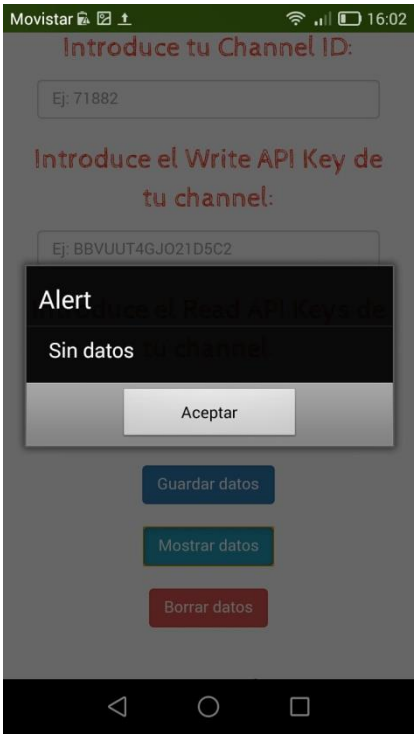

Figura 1.31. Tras pulsar el botón mostrar datos se muestra un Alert con el contenido "sin datos" si en memoria local no hay almacenado ningún valor

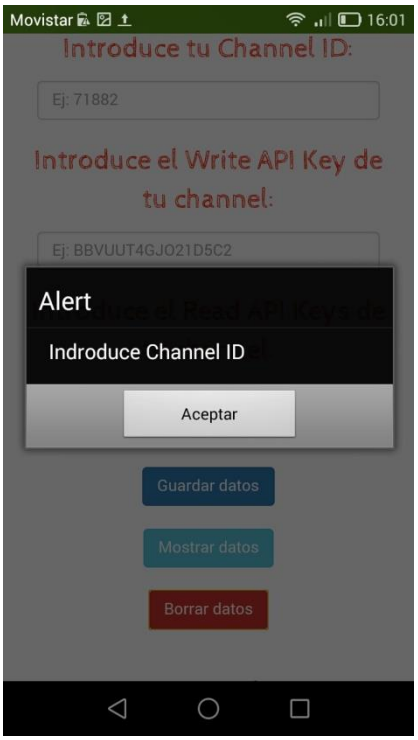

Figura 1.32. Tras pulsar el botón de borrar datos se muestra un Alert con el contenido "Introduce Channel ID"

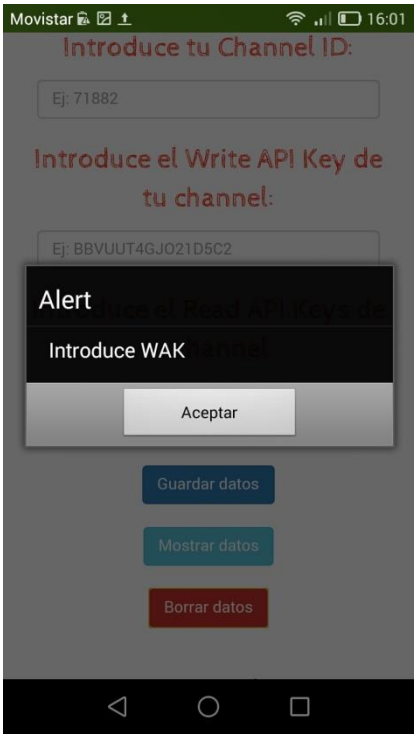

Figura 1.33. Tras pulsar el botón de borrar datos se muestra un Alert con el contenido "Introduce WAK" (Introduce Write API Key)

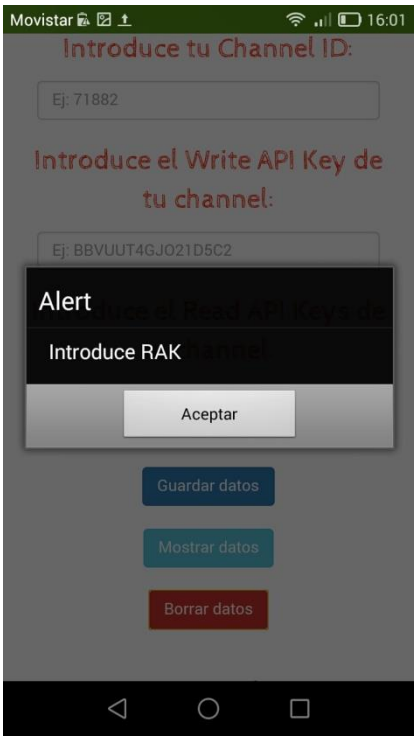

Figura 1.34. Tras pulsar el botón de borrar datos se muestra un Alert con el contenido "Introduce RAK" (Introduce Read API Keys)

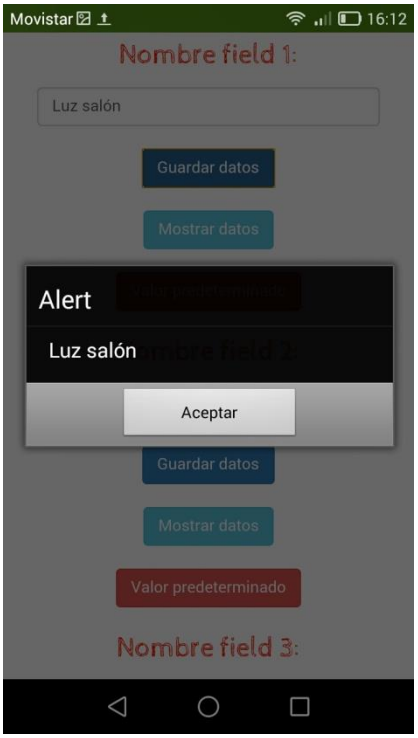

Figura 1.35. Tras pulsar el botón de guardar datos en los campos field 1 a field 8 se muestra en un Alert el resultado guardado

Figura 1.36. Tras pulsar el botón de mostrar datos en los campos field 1 a field 8 se muestra en un Alert el resultado guardado o si no hay datos guardados muestra de "Field 1" a Field 8"

Figura 1.37. Tras pulsar el botón de valor predeterminado de los campos field 1 a field 8 se muestra en un Alert "Valor por defecto restablecido"

Una vez explicado el entorno gráfico del bloque de configuración y las funcionalidades que aportan pasamos a detallar la implementación de dichas funcionalidades.

Dentro del script de funcionalidades (Código 1.6) hemos creado una serie de variables que serán utilizadas a lo largo del script con distintas funcionalidades (Código 1.7).

```
var channel_id;
var write_api_key;
var read_api_key;
var datos0;
var datos1;
var datos2;
var campo1;
var campo2;
var campo3;
var campo4;
var campo5;
var campo6;
var campo7;
var campo8;
var nombrecampo1;
var nombrecampo2;
var nombrecampo3;
var nombrecampo4;
var nombrecampo5;
var nombrecampo6;
```

```
var nombrecampo7;
var nombrecampo8;
var luz1;
var luz2;
var luz3;
var luz4;
var luz5;
var luz6;
var luz7;
var luz8;
```

Código 1.7. Variables utilizadas por el script principal de la aplicación

En el caso de los parámetros Channer ID, Write API Key, Read API Keys, nombre field 1, nombre field 2, nombre field 3, nombre field 4, nombre field 5, nombre field 6, nombre field 7 y nombre field 8 (Figura 1.25 y Figura 1.27) las variables de las que hacen uso respectivamente son channel_id, write_api_key, read_api_key, nombrecampo1, nombrecampo2, nombrecampo3, nombrecampo4, nombrecampo5, nombrecampo6, nombrecampo7 y nombrecampo8. En estas variables almacenaremos el valor escrito en los formularios HTML5 disponibles para el Channel ID, Write API Key, Read API Keys y los nombre de los campos del uno al ocho (Código 1.8 y Código 1.9)

```
<p class="centrarobjeto letrasensores">Introduce tu Channel ID:</p>
<div class="container"><input type="number" class="form-control"
id="channel_id_htm" placeholder="Ej: 71882"></div>
<p class="centrarobjeto letrasensores">Introduce el Write API Key de tu
channel:</p>
<div class="container"><input type="text" class="form-control"
id="write_api_key_htm" placeholder="Ej: BBVUUT4GJO21D5C2"></div>
<p class="centrarobjeto letrasensores">Introduce el Read API Keys de tu
channel:</p>
<div class="container"><input type="text" class="form-control"
id="read_api_key_htm" placeholder="Ej: FYU7H1REB5VG7LI2"></div>

<div class="centrarobjeto"><button id="1" class="btn btn-
primary">Guardar datos</button></div><br>
<div class="centrarobjeto"><button id="2" class="btn btn-info">Mostrar
datos</button></div><br>
<div class="centrarobjeto"><button id="3" class="btn btn-danger">Borrar
datos</button></div>
```

Código 1.8. HTML5 de los formularios para Channel ID, Write API Key, Read API Keys y de los botones a pulsar para ejecutar instrucciones de guardado, mostrado o borrado

```
<p class="centrarobjeto letrasensores">Nombre field 1:</p>
<div class="container"><input type="text" class="form-control"
id="nombrecampo1_htm" placeholder="Ej: Luz salón 1"></div><br>
<div class="centrarobjeto"><button id="4" class="btn btn-
primary">Guardar datos</button></div><br>
<div class="centrarobjeto"><button id="5" class="btn btn-info">Mostrar
datos</button></div><br>
```

```
<div class="centrarobjeto"><button id="6" class="btn btn-danger">Valor predeterminado</button></div>
```

Código 1.9. HTML5 del formulario del campo "Nombre field 1" y de los botones a pulsar para ejecutar instrucciones de guardado, mostrado o restablecimiento de valor predeterminado

El efecto de guardado para los parámetro Channel ID, Write API Key y Read API Keys (Código 1.10) tiene lugar al presionar el botón de guardado (Figura 1.26) que está asociado a un evento en JQUERY que hace que se ejecute la instrucción de almacenar en las variables channel_id, write_api_key y read_api_key el valor introducido en el formulario para posteriormente almacenar dicho valor en una variable en memoria local haciendo uso de la función localStorage. También mediante el uso de la función localStorage podremos volver a visualizar el valor almacenado en memoria local, para el caso del botón mostrar datos (Figura 1.26) también asociado a un evento en JQUERY, y podremos borrar dicho valor almacenado en memoria local pulsando sobre el botón borrar datos (Figura 1.26) también asociado a otro evento en JQUERY.

```
$("body").on("click","#1",function() {
      //alert($("#channel_id").val());
      channel_id = $("#channel_id_htm").val(); //id del canal
      write_api_key = $("#write_api_key_htm").val();
      read_api_key = $("#read_api_key_htm").val();
      //alert(write_api_key);
      localStorage.setItem('key0', channel_id);
      localStorage.setItem('key1', write_api_key);
      localStorage.setItem('key2', read_api_key);
      datos0 = localStorage.getItem('key0');
      datos1 = localStorage.getItem('key1');
      datos2 = localStorage.getItem('key2');
      alert(datos0);
      alert(datos1);
      alert(datos2);
      });
$("body").on("click","#2",function() {
      datos0 = localStorage.getItem('key0');//channelid
      datos1 = localStorage.getItem('key1');//writeapikey
      datos2 = localStorage.getItem('key2');//readapikey
            if
(((localStorage.getItem('key0'))==null)&&((localStorage.getItem('key1'))
==null)&&((localStorage.getItem('key2'))==null)) {alert("Sin datos");}
                  else {alert(datos0);
                        alert(datos1);
                        alert(datos2);}//Mostrar CID,WAK, RAK
      });
$("body").on("click","#3",function() {
      datos0 = localStorage.removeItem('key0');
      alert("Indroduce Channel ID");
      datos1 = localStorage.removeItem('key1');
      alert("Introduce WAK");
      datos2 = localStorage.removeItem('key2');
      alert("Introduce RAK");
```

```
        });
```

Código 1.10. Funciónes en JQUERY que permiten almacenar datos en memoria local, leer los datos almacenados o borrarlos. Para Channel ID, Write API Key y Read API Keys

En cuanto a la personalización del nombre de los campos se procede a explicar únicamente el primer campo ya que el resto de los campos funcionan exactamente igual. El efecto de guardado del nuevo nombre del campo field 1 (Código 1.11) tiene lugar al presionar el botón de guardado (Figura 1.27) que está asociado a un evento en JQUERY que hace que se ejecute la instrucción de almacenar en la variable nombrecampo1 el valor introducido en el formulario para posteriormente almacenar dicho valor en una variable en memoria local haciendo uso de la función localStorage. También mediante el uso de la función localStorage podremos volver a visualizar el valor almacenado en memoria local, para el caso del botón mostrar datos (Figura 1.27) también asociado a un evento en JQUERY, y podremos borrar y restablecer dicho valor almacenado en memoria local pulsando sobre el botón valor predeterminado (Figura 1.27) también asociado a otro evento en JQUERY.

```
$("body").on("click","#4",function() {
      nombrecampo1 = $("#nombrecampo1_htm").val(); //Guardar nombre que
ponemos a field 1
      localStorage.setItem('key3', nombrecampo1);
      alert(localStorage.getItem('key3'));
      });
$("body").on("click","#5",function() {
      if ((localStorage.getItem('key3'))==null) {alert("Field 1");}
            else {alert(localStorage.getItem('key3'));}//Mostrar nombre
de field 1
      });
$("body").on("click","#6",function() {
      localStorage.removeItem('key3');//Valor predeterminado de field 1
      nombrecampo1 = "Field 1"; //Field 1 es el valor por defecto
      alert("Valor por defecto restablecido");
      });
if (localStorage.getItem('key3') == null) {nombrecampo1 = "Field 1";}
      else {nombrecampo1 = localStorage.getItem('key3');}
```

Código 1.11 Funciónes en JQUERY que permiten almacenar datos en memoria local, leer los datos almacenados o restablecer un valor determinado. Para campo "Nombre field 1"

Damos por finalizada la explicación del primer bloque donde hemos tratado el entorno gráfico de la configuración de la aplicación mediante los tres parámetros de configuración llamados Channel ID, Write API Key y Read API Keys, además de la opción de personalización del nombre de los campos para poder identificar nuestros sensores en la aplicación de una forma sencilla y la implementación de todas sus funcionalidades asociadas.

Pasamos por al segundo bloque donde se detalla el entorno gráfico de los sensores, la opción de refresco de los datos que se muestran en la aplicación y sus funcionalidades asociadas (Figura 1.38, Figura 1.39 y Figura 1.40). Dentro del campo mis sensores encontramos la siguiente interfaz:

- Actualizar datos: Botón que al ser pulsado activa un evento en JQUERY que ejecuta una función que permite, a través de peticiones HTTP, obtener el último valor de los campos del canal creado en ThingSpeak para posteriormente representarlos gráficamente en la aplicación.

- Switches Luz: Muestra mediante HTML5 y CSS si los interruptores de luz configurados en ThingSpeak se encuentran encendidos o apagados. Un valor de cero es apagado mientras que un valor de uno es encendido. Permite, pulsando sobre el interruptor de la aplicación, apagar o encender la luz enviando una petición HTTP para que se produzca la modificación en la base de datos.

- Regulador Luz: Muestra mediante HTML5 y CSS el nivel al que se encuentran los reguladores des de luz configurados en ThingSpeak variando desde el cero por cien hasta el cien por cien. Permite, modficar el valor del regulador de la aplicación y, por tanto, al pulsar después sobre el botón de actualización (Update to X%) permite variar el tanto por cien del regulador de luz enviando una petición HTTP para que se produzca la modificación en la base de datos.

- Intensidad Luminosa: Muestra mediante HTML5 y CSS el valor más reciente de la intensidad luminosa procedente del sensor y que se encuentra almacenado en la base de datos.

- Temperatura: Muestra mediante HTML5 y CSS el valor más reciente de la temperatura procedente del sensor y que se encuentra almacenado en la base de datos.

- Humedad: Muestra mediante HTML5 y CSS el valor más reciente de la humedad almacenado en la base de datos que procede del sensor.

- Lluvia: Muestra mediante HTML5 y CSS el valor más reciente indicado por el sensor de lluvia y que está almacenado en la base de datos. Este valor puede ser un cero lo cual indica que no llueve, la aplicación muestra un “Lluvia: NO”, o puede ser un uno lo cual indica sí llueve, la aplicación muestra un “Lluvia: SÍ”.

Figura 1.38. Zona mis sensores, primera parte

Figura 1.39. Zona mis sensores, segunda parte

Figura 1.40. Zona mis sensores, tercera parte

El número máximo de sensores mencionados anteriormente que se pueden configurar es de ocho, pueden ser todos del mismo tipo como por ejemplo reguladores de luz o pueden ser una mezcla de varios tipos.

Una vez explicado el entorno gráfico de los sensores y las funcionalidades que aportan pasamos a detallar la implementación de dichas funcionalidades.

Comenzamos explicando el botón de actualización de los datos (Figura 1.38). El efecto de actualización de los campos switches luz, regulador luz, intensidad luminosa, temperatura, humedad y lluvia (Código 1.12) tiene lugar al presionar el botón de actualización de los datos el cual está asociado a un evento en JQUERY que hace que se ejecute la instrucción de leer y almacenar en ocho variables temporales los ocho valores de los campos del canal para poder realizar las actualizaciones de dichos valores en la aplicación. Mediante una petición HTTP con el método GET se solicita el último valor de cada uno de los ocho campos (Código 1. 12) que se guardarán en las variables campo1, campo2, campo3, campo4, campo5, campo6, campo7 y campo8 (Código 1.7).

```
$get.("
https://api.thingspeak.com/channels/"+localStorage.getItem('key0')+"
/fields/1/last.json?api_key="+localStorage.getItem('key2'),function(resu
lt)  {
```

```
        campo1 = result.field1;
      });
```

Código 1.12. Petición HTTP con el método GET para obtener valores más actuales del campo 1 del canal

Tras obtener mediante la petición HTTP los valores de los campos se procede a ejecutar un borrado de todo el contenido HTML5 de los campos que hay en la aplicación por si se ha producido algún cambio en la configuración de los sensores de ThingSpeak. Para ello se utiliza la función JQUERY empty() aplicada sobre un identificador que permite eliminar el contenido HTML5 que haya en la etiqueta que contiene dicho identificador (Código 1.13).

```
$("#idlight_s0").empty();
...
$("#idlight_s7").empty();

$("#idlight_r0").empty();
...
$("#idlight_r7").empty();

$("#idlight_i0").empty();
...
$("#idlight_i7").empty();

$("#idtemp0").empty();
...
$("#idtemp7").empty();

$("#idhum0").empty();
...
$("#idhum7").empty();

$("#idrain0").empty();
...
$("#idrain7").empty();
```

Código 1.13. Pseudocódigo de la parte de código que elimina los contenidos HTML5 que se encuentran dentro de la etiqueta marcada por el identificador

Por último, se modifican mediante nuevo código HTML5 que se inserta dentro del código HTML5 ya creado mediante funciones JQUERY que hacen uso de los identificadores de las etiquetas referentes a los campos (Código 1.14), los valores ya actualizados que se van a visualizar de los campos switches luz, regulador luz, intensidad luminosa, temperatura, humedad y lluvia.

```
<div class="centrarobjeto letrasensores">Switches Luz:</div>
<span id="idlight_s0"></span>
<span id="idlight_s1"></span>
<span id="idlight_s2"></span>
<span id="idlight_s3"></span>
```

```
<span id="idlight_s4"></span>
<span id="idlight_s5"></span>
<span id="idlight_s6"></span>
<span id="idlight_s7"></span>
<br>
<div class="centrarobjeto letrasensores">Regulador Luz:</div>
<span id="idlight_r0"></span>
<span id="idlight_r1"></span>
<span id="idlight_r2"></span>
<span id="idlight_r3"></span>
<span id="idlight_r4"></span>
<span id="idlight_r5"></span>
<span id="idlight_r6"></span>
<span id="idlight_r7"></span>
<br>
<div class="centrarobjeto letrasensores">Intensidad Luminosa:</div>
<span id="idlight_i0"></span>
<span id="idlight_i1"></span>
<span id="idlight_i2"></span>
<span id="idlight_i3"></span>
<span id="idlight_i4"></span>
<span id="idlight_i5"></span>
<span id="idlight_i6"></span>
<span id="idlight_i7"></span>
<br>
<div class="centrarobjeto letrasensores">Temperatura:</div>
<span id="idtemp0"></span>
<span id="idtemp1"></span>
<span id="idtemp2"></span>
<span id="idtemp3"></span>
<span id="idtemp4"></span>
<span id="idtemp5"></span>
<span id="idtemp6"></span>
<span id="idtemp7"></span>
<br>
<div class="centrarobjeto letrasensores">Humedad:</div>
<span id="idhum0"></span>
<span id="idhum1"></span>
<span id="idhum2"></span>
<span id="idhum3"></span>
<span id="idhum4"></span>
<span id="idhum5"></span>
<span id="idhum6"></span>
<span id="idhum7"></span>
<br>
<div class="centrarobjeto letrasensores">Lluvia:</div>
<span id="idrain0"></span>
<span id="idrain1"></span>
<span id="idrain2"></span>
<span id="idrain3"></span>
<span id="idrain4"></span>
<span id="idrain5"></span>
<span id="idrain6"></span>
<span id="idrain7"></span>
```

Código 1.14. Nombre de los identificadores que se utilizan en la función JQUERY para insertar nuevo código HTML5 sobre el código ya existente

Antes de adentrarnos en la actualización de cada uno de los campos hay que mencionar que para saber con qué campos estamos trabajando, es decir, qué campos se han configurado en ThingSpeak y por tanto saber cómo seleccionar dichos campos configurados para que sean representados en la aplicación, se ha hecho uso de una petición HTTP con el método GET para conocer los campos configurados en el entorno de ThingSpeak y a continuación mediante ocho funciones switch() se ha procedido a seleccionar dentro de cada campo qué tipo de sensor tiene configurado si es un switch de luz, un regulador de luz, un sensor de intensidad luminosa, un sensor de temperatura, un sensor de humedad o un sensor de lluvia para que esté perfectamente asociado con la configuración del canal de ThingSpeak (Código 1.15).

```
$get.("
https://api.thingspeak.com/channels/"+localStorage.getItem('key0')+"
/feeds.json?api_key="+localStorage.getItem('key2'),function(result)  {
      //CANAL 1
      switch(result.channel.field1){case "light_s0" ... case "rain7"}
      //CANAL 2
      switch(result.channel.field2){case "light_s0" ... case "rain7"}
      //CANAL 3
      switch(result.channel.field3){case "light_s0" ... case "rain7"}
      //CANAL 4
      switch(result.channel.field4){case "light_s0" ... case "rain7"}
      //CANAL 5
      switch(result.channel.field5){case "light_s0" ... case "rain7"}
      //CANAL 6
      switch(result.channel.field6){case "light_s0" ... case "rain7"}
      //CANAL 7
      switch(result.channel.field7){case "light_s0" ... case "rain7"}
      //CANAL 8
      switch(result.channel.field8){case "light_s0" ... case "rain7"}
     });
```

Código 1.15. Pseudocódigo de la petición HTTP con el método GET para obtener el nombre de los campos configurados en ThingSpeak y funciones swtich() que seleccionan qué tipo de sensor tiene cada campo y su actualización en la interfaz gráfica de la aplicación

Dentro de los switches de luz existen dos posibilidades: la actualización del valor del interruptor en la base de datos debido a una petición HTTP procedente de la Raspberry o la actualización del valor del interruptor en la base de datos desde una petición HTTP procedente de la aplicación móvil. Para el caso de la petición HTTP procedente de la Raspberry, se ha de actualizar el valor en la aplicación móvil. Para ello, una vez realizado el proceso de seleccionar qué tipo de sensor tiene cada campo (Código 1.15) se realiza la inclusión del nuevo código HTML5 referente al interruptor de luz indicando si está apagado o encendido (Código 1.16).

```
$("#idlight_s0").html('<div class="centrarobjeto
letracampos">'+nombrecampo2+'</div><div class="switch"><input id="cmn-
```

```
toggle-1-field2" class="cmn-toggle cmn-toggle-round"
type="checkbox"><label for="cmn-toggle-1-field2"></label></div>');
      if (campo2=="1"){$("#cmn-toggle-1-field2").prop('checked',
true);}
      else if(campo2=="0"){$("#cmn-toggle-1-field2").prop('checked',
false);}
```

Código 1.16. Inclusión del nuevo código HTML5 para el caso de un interruptor de luz

Para el caso donde la aplicación móvil tiene como fin modificar el valor del interruptor de luz que se encuentra en la base de datos, se envía una petición HTTP con el método POST que interactúa contra la REST API de ThingSpeak permitiendo modificar el valor del interruptor que hay en la base de datos. Esta secuencia se produce al pulsar sobre el interruptor de luz que hace que se active un evento JQUERY que permite que se ejecute el código que envía la petición HTTP (Código 1.17).

```
$("body").on("click","#cmn-toggle-1-field1",function() {
      if($("#cmn-toggle-1-field1").is(":checked")==true){
      $.post("https://api.thingspeak.com/update.json?api_key="+localSto
rage.getItem('key1')+"&field1=1");}
      if($("#cmn-toggle-1-field1").is(":checked")==false){
      $.post("https://api.thingspeak.com/update.json?api_key="+localSto
rage.getItem('key1')+"&field1=0");}
     });
```

Código 1.17.Petición HTTP mediante el método POST para actualizar un interruptor de luz en la base de datos desde la aplicación móvil

Dentro de los reguladores de luz también tenemos dos posibilidades: la actualización del valor del regulador en la base de datos debido a una petición HTTP procedente de la Raspberry o la actualización del valor del regulador en la base de datos desde una petición HTTP procedente de la aplicación móvil. Para el caso de la petición HTTP procedente de la Raspberry, se ha de actualizar el valor en la aplicación móvil. Para ello, una vez realizado el proceso de seleccionar qué tipo de sensor tiene cada campo (Código 1.15) se realiza la inclusión del nuevo código HTML5 referente al regulador de luz indicando si está apagado o encendido (Código 1.18).

```
$("#idlight_r0").html('<div class="centrarobjeto
letracampos">'+nombrecampo2+'</div><input id="regulador-1-field2"
type="range" min="0" max="100" value="'+campo2+'"><br><div
class="centrarobjeto"><button class="btn btn-primary valorsensoresr"
id="regulador-1-field2-button">Update to <span id="idregulador-1-
field2">'+campo2+'</span>%</button></div><br>');
```

Código 1.18 Inclusión del nuevo código HTML5 para el caso de un regulador de luz

Para el caso donde la aplicación móvil tiene como fin modificar el valor del regulador de luz que se encuentra en la base de datos, se envía una petición HTTP con el método POST que interactúa contra la REST API de ThingSpeak permitiendo modificar el valor del interruptor que hay en la base de datos. Esta secuencia se produce al pulsar sobre el botón de actualización de los reguladores de luz (Update to X%), previamente habiendo variado el valor del regulador de luz, que hace que se active un evento JQUERY que permite que se ejecute el código que envía la petición HTTP (Código 1.19).

```
$("body").on("change","#regulador-1-field1",function() {
      luz1 = $("#regulador-1-field1").val();
      $("#idregulador-1-field1").html(luz1);
      });
$("body").on("click","#regulador-1-field1-button",function() {
$.post("https://api.thingspeak.com/update.json?api_key="+localStorage.ge
tItem('key1')+"&field1="+luz1);
      });
```

Código 1.19. Petición HTTP mediante el método POST para actualizar un regulador de luz en la base de datos desde la aplicación móvil

En cuanto a los sensores de intensidad luminosa también se realiza la inclusión del nuevo código HTML5 una vez realizado el proceso de seleccionar qué tipo de sensor tiene cada campo (Código 1.20).

```
$("#idlight_i0").html('<div class="centrarobjeto
letracampos">'+nombrecampo1+'</div><p class="centrarobjeto
valorsensoresi">'+campo1+" Cd"+"</p>");
```

Código 1.20. Inclusión del nuevo código HTML5 para el caso de los sensores de intensidad luminosa

En referencia a los sensores de temperatura también se realiza la inclusión del nuevo código HTML5 una vez realizado el proceso de seleccionar qué tipo de sensor tiene cada campo (Código 1.21).

```
$("#idtemp0").html('<div class="centrarobjeto
letracampos">'+nombrecampo1+'</div><p class="centrarobjeto
valorsensorest">'+campo1+" °C"+"</p>");
```

Código 1.21. Inclusión del nuevo código HTML5 para el caso de los sensores de temperatura

Respecto a los sensores de humedad también se realiza la inclusión del nuevo código HTML5 una vez realizado el proceso de seleccionar qué tipo de sensor tiene cada campo (Código 1.22).

```
$("#idhum0").html('<div class="centrarobjeto
letracampos">'+nombrecampo1+'</div><p class="centrarobjeto
valorsensoresh">'+campo1+"% RH"+"</p>");
```

Código 1.22. Inclusión del nuevo código HTML5 para el caso de los sensores de humedad

En cuanto a los sensores de lluvia también se realiza la inclusión del nuevo código HTML5 una vez realizado el proceso de seleccionar qué tipo de sensor tiene cada campo (Código 1.23).

```
if(campo1=="0"){
      $("#idrain0").html('<div class="centrarobjeto
letracampos">'+nombrecampo1+'</div><p class="centrarobjeto
valorsensoresl">NO</p>');}
else if(campo1=="1"){
      $("#idrain0").html('<div                    class="centrarobjeto
letracampos">'+nombrecampo1+'</div><p              class="centrarobjeto
valorsensoresl">SÍ</p>');}
```

Código 1.23. Inclusión del nuevo código HTML5 para el caso de los sensores de lluvia

Por último, hay que destacar que al cada vez que se ejecuta la aplicación, siempre que la aplicación estuviera cerrada previamente, se ejecuta una única vez de forma automática la función de actualización de los valores de los campos en la interfaz gráfica de la aplicación (Código 1.15).

Damos por finalizada la explicación del segundo donde hemos tratado el entorno gráfico de los sensores, la opción de refresco de los datos que se muestran en la aplicación, las funcionalidades que son llevadas a cabo en este entorno gráfico y finalmente la explicación de la implementación de todas las funcionalidades tratadas.

Ahora que ya conocemos el funcionamiento interno de la aplicación de domótica "Home Smart Home" pasamos a detallar el entorno del desarrollo web para emular la comunicación entre sensores, Raspberry y base de datos donde trataremos el entorno gráfico del desarrollo web, sus funcionalidades y la implementación de dichas funcionalidades.

1.7 Entorno del desarrollo web para emular la comunicación entre sensores, Raspberry y base de datos

En este apartado se pasa a detallar el entorno gráfico del desarrollo web, sus funcionalidades y la implementación de dichas funcionalidades. Para ello primero explicaremos cómo es el entorno gráfico y cuáles son sus funcionalidades y posteriormente analizaremos la implementación de dichas funcionalidades. Cabe mencionar que este desarrollo web ha sido diseñado específicamente para que en el campo1, campo7 y campo8 haya interruptores luz, en el campo 2 haya un regulador de luz, en el campo 3 un sensor de intensidad luminosa, en el campo 4 un sensor de temperatura, en el campo 5 un sensor de humedad y en el campo 6 un sensor de lluvia. Todos estos interruptores, reguladores y sensores del desarrollo web emulan la parte del sistema End-to-End formada por interruptores, reguladores, sensores y Raspberry, que actualizan la base de datos de ThingSpeak periódicamente para poder simular el comportamiento del sistema completo.

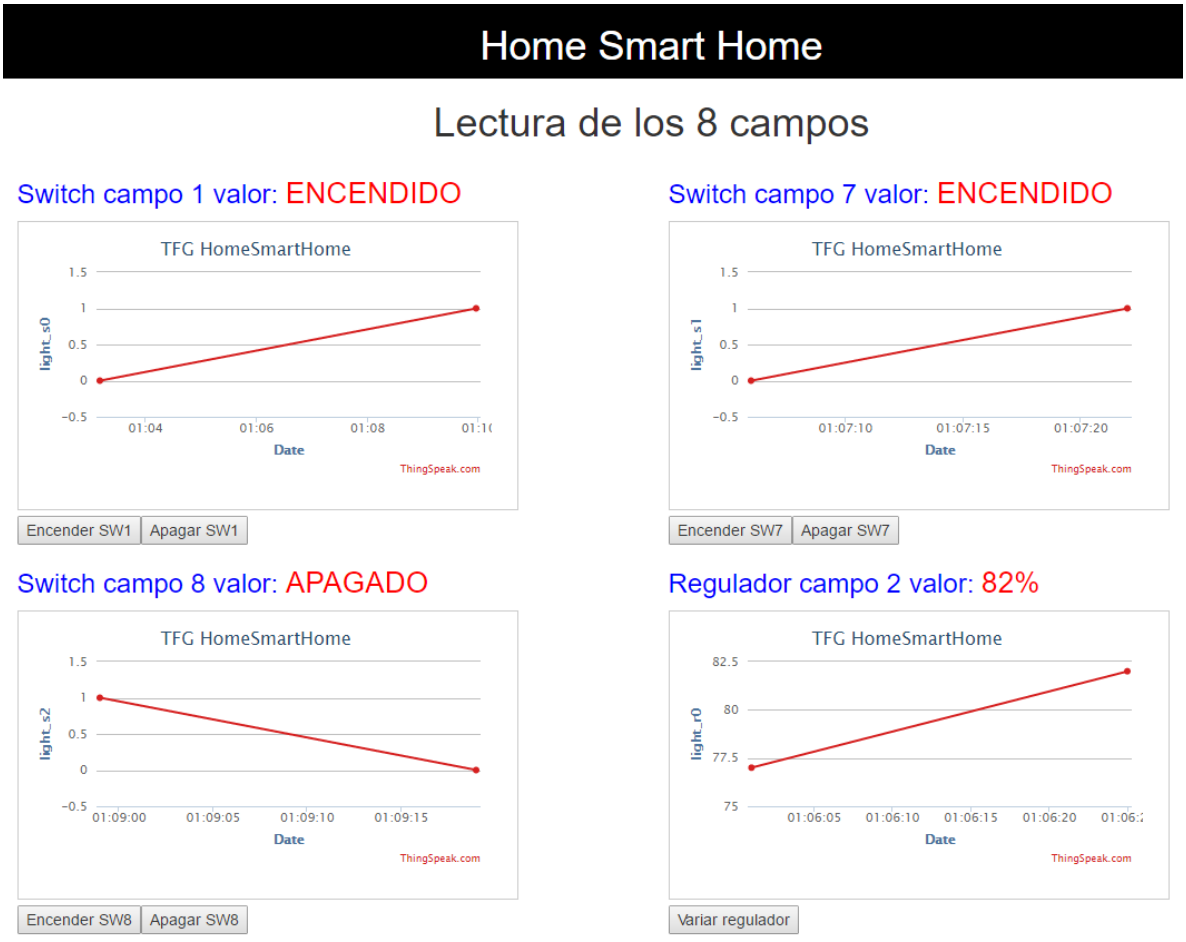

Figura 1.41. Entorno gráfico del desarrollo web, primera parte

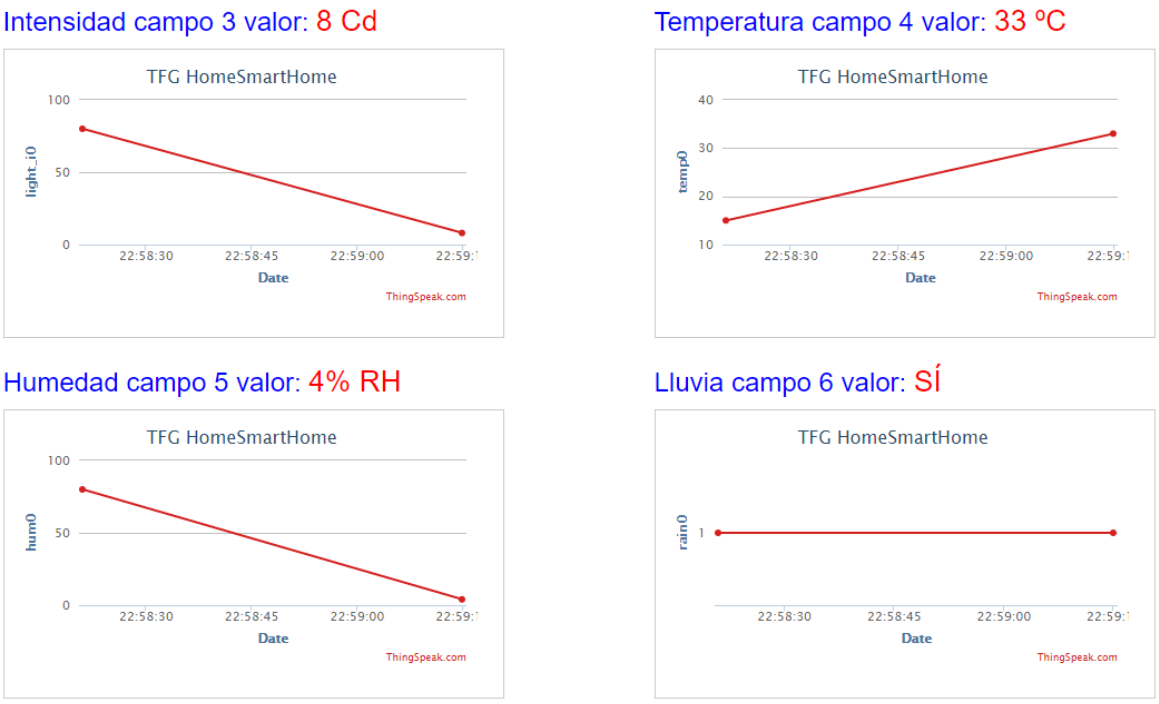

Figura 1.42. Entorno gráfico del desarrollo web, segunda parte

Los elementos del entorno gráfico del desarrollo web (Figura 1.41 y Figura 1.42) con los que contamos son los siguientes:

- Switch campo 1: Muestra el valor del interruptor que se encuentra en el campo 1. Este valor puede ser encendido o apagado. Contiene un botón de encendido y otro de apagado que mediante peticiones HTTP con el método POST actualizan el valor del campo 1 en la base de datos. Cada diez segundos se envía una petición HTTP con el método GET para actualizar el valor mostrado en el entorno gráfico del desarrollo web del campo 1.

- Switch campo 7: Muestra el valor del interruptor que se encuentra en el campo 7. Este valor puede ser encendido o apagado. Contiene un botón de encendido y otro de apagado que mediante peticiones HTTP con el método POST actualizan el valor del campo 7 en la base de datos. Cada diez segundos se envía una petición HTTP con el método GET para actualizar el valor mostrado en el entorno gráfico del desarrollo web del campo 7.

- Switch campo 8: Muestra el valor del interruptor que se encuentra en el campo 8. Este valor puede ser encendido o apagado. Contiene un botón de encendido y otro de apagado que mediante peticiones HTTP con el método POST actualizan el valor del campo 8 en la base de datos. Cada diez segundos se envía una petición HTTP con el método GET para actualizar el valor mostrado en el entorno gráfico del desarrollo web del campo 8.

- Regulador campo 2: Muestra el valor del regulador que se encuentra en el campo 2. Este valor varía del cero (apagado) al cien (máxima corriente eléctrica y por tanto máxima luminosidad). Contiene un botón de variación del regulador que modifica aleatoriamente el valor del regulador entre los valores comprendido entre cero y cien. Cada diez segundos se envía una petición HTTP con el método GET para actualizar el valor mostrado en el entorno gráfico del desarrollo web del campo 2.

- Intensidad campo 3: Muestra el valor del de la intensidad luminosa en Candelas del campo 3. Cada sesenta segundos se envía una petición HTTP con el método POST para actualizar el valor del campo 3 en la base de datos. Cada diez segundos se envía una petición HTTP con el método GET para actualizar el valor mostrado en el entorno gráfico del desarrollo web del campo 3.

- Temperatura campo 4: Muestra el valor del de la temperatura en Grados Centígrados del campo 4. Cada sesenta segundos se envía una petición HTTP con el método POST para actualizar el valor del campo 4 en la base de datos. Cada diez segundos se envía una petición HTTP con el método GET para actualizar el valor mostrado en el entorno gráfico del desarrollo web del campo 4.

- Humedad campo 5: Muestra el valor del de la humedad en tanto por cien de Humedad Relativa del campo 5. Cada sesenta segundos se envía una petición HTTP con el método POST para actualizar el valor del campo 5 en la base de datos. Cada diez segundos se envía una petición HTTP con el método GET para actualizar el valor mostrado en el entorno gráfico del desarrollo web del campo 5.

- Lluvia campo 6: Muestra el valor del sensor de lluvia del campo 6. Este valor puede ser sí (si llueve) o no (si no llueve). Cada 60 segundos se envía una petición HTTP con el método POST para actualizar el valor del campo 6 en la base de datos. Cada 10 segundos se envía una petición HTTP con el método GET para actualizar el valor mostrado en el entorno gráfico del desarrollo web del campo 6.

Una vez explicado el entorno gráfico del desarrollo web y las funcionalidades que aportan pasamos a detallar la implementación de dichas funcionalidades. Hay cuatro clases de funciones principales que se van a tratar: la primera es la lectura de la base de datos nada más abrir la página web para poder mostrar el último valor almacenado en la base de datos, la segunda es la lectura cada diez segundos de la base de datos para poder mostrar su último valor en la página web de forma periódica, la tercera es el envío de nuevos datos a la base de datos de sensores de intensidad luminosa, temperatura, humedad y lluvia cada

sesenta segundos para poder comprobar su cambio en la aplicación móvil y la cuarta es el envío de nuevos datos a la base de datos de interruptores y reguladores de luz cada vez que se pulse el botón disponible en la página web de forma que podamos comprobar su variación en la aplicación móvil.

La primera funcionalidad hace uso de peticiones HTTP con el método GET para obtener los últimos valores de los sensores que han sido almacenados en la base de datos (Código 1.24). Sólo se ejecuta una vez al abrir la página web.

```
setTimeout(function(){
//Leo y almaceno en una variable temporal los 8 campos para poder
realizar las actualizaciones de los sensores en la página web
$.get("https://api.thingspeak.com/channels/71882/fields/1/last.json?api_
key=FYU7H1REB5VG7LI2",function(result){
      campo1 = result.field1;
//CANAL 1 switch
      if(campo1=="0"){$("#idlight_s0").html('APAGADO');}
      else if(campo1=="1"){$("#idlight_s0").html('ENCENDIDO');}
      });
$.get("https://api.thingspeak.com/channels/71882/fields/2/last.json?api_
key=FYU7H1REB5VG7LI2",function(result){
      campo2 = result.field2;
//CANAL 2 regulador
      $("#idlight_r0").html(campo2+"%");
      });
$.get("https://api.thingspeak.com/channels/71882/fields/3/last.json?api_
key=FYU7H1REB5VG7LI2",function(result){
      campo3 = result.field3;
//CANAL 3 intensidad
      $("#idlight_i0").html(campo3+" Cd");
      });
$.get("https://api.thingspeak.com/channels/71882/fields/4/last.json?api_
key=FYU7H1REB5VG7LI2",function(result){
      campo4 = result.field4;
//CANAL 4 temperatura
      $("#idtemp0").html(campo4+" °C");
      });
$.get("https://api.thingspeak.com/channels/71882/fields/5/last.json?api_
key=FYU7H1REB5VG7LI2",function(result){
      campo5 = result.field5;
//CANAL 5 humedad
      $("#idhum0").html(campo5+"% RH");
      });
$.get("https://api.thingspeak.com/channels/71882/fields/6/last.json?api_
key=FYU7H1REB5VG7LI2",function(result){
      campo6 = result.field6;
//CANAL 6 lluvia
      if(campo6=="0"){$("#idrain0").html('NO');}
      else if(campo6=="1"){$("#idrain0").html('SÍ');}
      });
$.get("https://api.thingspeak.com/channels/71882/fields/7/last.json?api_
key=FYU7H1REB5VG7LI2",function(result){
      campo7 = result.field7;
```

```
//CANAL 7 switch
      if(campo7=="0"){$("#idlight_s1").html('APAGADO');}
      else if(campo7=="1"){$("#idlight_s1").html('ENCENDIDO');}
      });
$.get("https://api.thingspeak.com/channels/71882/fields/8/last.json?api_
key=FYU7H1REB5VG7LI2",function(result){
      campo8 = result.field8;
//CANAL 8 switch
      if(campo8=="0"){$("#idlight_s2").html('APAGADO');}
      else if(campo8=="1"){$("#idlight_s2").html('ENCENDIDO');}
      });
});
```

Código 1.24. Peticiones HTTP con el método GET para obtener los últimos valores de los sensores que han sido almacenados en la base de datos. Se ejecuta una única vez

La segunda funcionalidad es idéntica a la primera con la diferencia que se ejecuta cada diez segundos de forma que se mantiene actualizada la información del desarrollo web (Código 1.25)

```
setTimeout(function(){
//Leo y almaceno en una variable temporal los 8 campos para poder
realizar las actualizaciones de los sensores en la página web

$.get("https://api.thingspeak.com/channels/71882/fields/1/last.json?api
_key=FYU7H1REB5VG7LI2",function(result){
      campo1 = result.field1;
//CANAL 1 switch
      if(campo1=="0"){$("#idlight_s0").html('APAGADO');}
      else if(campo1=="1"){$("#idlight_s0").html('ENCENDIDO');}
      });
$.get("https://api.thingspeak.com/channels/71882/fields/2/last.json?api
_key=FYU7H1REB5VG7LI2",function(result){
      campo2 = result.field2;
//CANAL 2 regulador
      $("#idlight_r0").html(campo2+"%");
      });
$.get("https://api.thingspeak.com/channels/71882/fields/3/last.json?api
_key=FYU7H1REB5VG7LI2",function(result){
      campo3 = result.field3;
//CANAL 3 intensidad
      $("#idlight_i0").html(campo3+" Cd");
      });
$.get("https://api.thingspeak.com/channels/71882/fields/4/last.json?api
_key=FYU7H1REB5VG7LI2",function(result){
      campo4 = result.field4;
//CANAL 4 temperatura
      $("#idtemp0").html(campo4+" °C");
      });
$.get("https://api.thingspeak.com/channels/71882/fields/5/last.json?api
_key=FYU7H1REB5VG7LI2",function(result){
      campo5 = result.field5;
//CANAL 5 humedad
      $("#idhum0").html(campo5+"% RH");
      });
$.get("https://api.thingspeak.com/channels/71882/fields/6/last.json?api
```

```
_key=FYU7H1REB5VG7LI2",function(result){
      campo6 = result.field6;
//CANAL 6 lluvia
      if(campo6=="0"){$("#idrain0").html('NO');}
      else if(campo6=="1"){$("#idrain0").html('SÍ');}
      });
$.get("https://api.thingspeak.com/channels/71882/fields/7/last.json?api
_key=FYU7H1REB5VG7LI2",function(result){
      campo7 = result.field7;
//CANAL 7 switch
      if(campo7=="0"){$("#idlight_s1").html('APAGADO');}
      else if(campo7=="1"){$("#idlight_s1").html('ENCENDIDO');}
      });
$.get("https://api.thingspeak.com/channels/71882/fields/8/last.json?api
_key=FYU7H1REB5VG7LI2",function(result){
      campo8 = result.field8;
//CANAL 8 switch
      if(campo8=="0"){$("#idlight_s2").html('APAGADO');}
      else if(campo8=="1"){$("#idlight_s2").html('ENCENDIDO');}
      });
},10000);
```

Código 1.25. Peticiones HTTP con el método GET para obtener los últimos valores de los sensores que han sido almacenados en la base de datos. Se ejecuta cada diez segundos

La tercera funcionalidad envía nuevos datos mediante una petición HTTP con el método POST cada sesenta segundos a la base de datos sobre los sensores de intensidad luminosa, temperatura, humedad y lluvia. Estos datos enviados son generados de forma aleatoria.

```
setInterval(function(){
$.post("https://api.thingspeak.com/update.json?api_key=BBVUUT4GJO21D5C2
&field3="+Math.round(Math.random()*100)+"&field4="+Math.round(Math.rand
om()*50)+"&field5="+Math.round(Math.random()*100)+"&field6="+Math.round
(Math.random())); //Campo 3
}, 60000);
```

Código 1.26. Peticiones HTTP con el método POST para actualizar de forma aleatoria el valor de los sensores de intensidad luminosa, temperatura, humedad y lluvia. Se ejecuta cada sesenta segundos

La cuarta funcionalidad envía nuevos datos de los tres interruptores de luz y del regulador de luz mediante peticiones HTTP con el método POST cada vez que se presiona uno de los botones creados para ello (Figura 1.41 y Código 1.27).

```
////Acción de cambiar DB en caso de pulsar sobre los botones
//campo1
$("body").on("click","#1",function() {
$.post("https://api.thingspeak.com/update.json?api_key=BBVUUT4GJO21D5C2&
field1=1"); //Enciendo
      });
$("body").on("click","#2",function() {
$.post("https://api.thingspeak.com/update.json?api_key=BBVUUT4GJO21D5C2&
field1=0"); //Apago
      });
```

```
//campo7
$("body").on("click","#3",function() {
$.post("https://api.thingspeak.com/update.json?api_key=BBVUUT4GJO21D5C2&
field7=1"); //Enciendo
      });
$("body").on("click","#4",function() {
$.post("https://api.thingspeak.com/update.json?api_key=BBVUUT4GJO21D5C2&
field7=0"); //Apago
      });
//campo8
$("body").on("click","#5",function() {
$.post("https://api.thingspeak.com/update.json?api_key=BBVUUT4GJO21D5C2&
field8=1"); //Enciendo
      });
$("body").on("click","#6",function() {
$.post("https://api.thingspeak.com/update.json?api_key=BBVUUT4GJO21D5C2&
field8=0"); //Apago
      });
//campo2
$("body").on("click","#7",function() {
$.post("https://api.thingspeak.com/update.json?api_key=BBVUUT4GJO21D5C2&
field2="+Math.round(Math.random()*100)); //Variar regulador
      });
```

Código 1.27. Peticiones HTTP con el método POST para actualizar el valor de los interruptores y el regulador de luz. Se ejecuta cada vez que se llama al evento JQUERY asociado a los botones de encendido y apagado de los interruptores y al botón variación del regulador de luz

Damos por finalizada la explicación del entorno gráfico del desarrollo web, sus funcionalidades y a la implementación de dichas funcionalidades.

1.8 Versión entregada

En este apartado se describe la versión de la aplicación que se ha entregado.

Se ha decidido compilar una única versión para dispositivos Android (versión 4.2) ya que únicamente se dispone de dispositivos Android para hacer uso de la aplicación. Aunque se ha compilado únicamente para dispositivos Android, la herramienta INTEL XDK permite realizar la compilación de la aplicación para que también pueda ser usada en dispositivos iOS, Windows Phone, Windows 8 y Windows 10.

Para el uso en plataformas Android se entrega el siguiente programa:

- Repositorio para descargar la versión 4.2 de la aplicación "Home Smart Home" con nombre el nombre HSH_TFG.apk: Link de descarga del repositorio de Google Drive

1.9 Conclusiones y trabajo futuro

Conclusiones

Podemos concluir que se ha conseguido desarrollar una aplicación de gestión de elementos básicos de la domótica como son los interruptores de luz, reguladores de luz, sensores de intensidad luminosa, sensores de temperatura, sensores de humedad y sensores de lluvia.

Como principales contribuciones podemos destacar:

1. Interfaz sencilla y agradable a la vista.

2. Fácil configuración y uso de la aplicación, en la cual sólo se deben introducir tres parámetros para que quede configurada y pueda comenzar a ser utilizada.

3. Modificación del nombre de cada campo, disponible para cada tipo de sensor, en la propia aplicación.

La configuración de dicha aplicación una vez descargada e instalada es realmente sencilla, teniendo tan solo que registrarnos en la web de ThingSpeak para la creación de un canal donde se van almacenando los datos de todos los sensores mencionados anteriormente para a continuación, mediante tres parámetros ofrecidos por la plataforma de ThingSpeak (Channel ID, Write API Key y Read API Keys) introducirlos en nuestra aplicación de forma que esta se auto-configure mostrándonos todos los datos de nuestros sensores y pudiendo interactuar con ellos desde cualquier lugar donde tengamos acceso a Internet.

Por todo ello, el desarrollo de esta aplicación la cual tiene un uso realmente sencillo proporciona un aumento de calidad de vida para aquellas personas que decidan utilizarla, permitiéndoles en todo momento gestionar su casa de una forma más centralizada lo que se traduce, en gran medida, en un ahorro económico.

Finalmente, la opción de extrapolación de la aplicación a nuevos requerimientos resulta llegar a ser bastante sencilla donde, por ejemplo, se pudiendo llegar a crear un sistema de alarma que nos permita verificar el estado de nuestra vivienda automático que avise a la policía directamente cuando se produce un robo en un domicilio o incluso un sistema de gestión de la vivienda que avise a los servicios sociales de forma automática si una persona mayor o discapacitada ha sufrido algún problema.

Ampliaciones futuras

La aplicación presentada puede ser ampliada y mejorada de muchas formas. El límite de mejora depende de los conocimientos y la originalidad del desarrollador. A continuación se describen una serie de mejoras que pueden ser implementadas para versiones futuras:

- Creación de nuevos parámetros a introducir en ThingSpeak y la aplicación que permitan configurar nuevas clases de sensores. A esta idea también le acompañan los desarrollos pertinentes para mostrar de forma gráfica los nuevos sensores en la aplicación.

- Configuración del interfaz de usuario de la aplicación a través del usuario final. Creación de unos desarrollos que permitan al usuario final personalizar la aplicación a su gusto.

- Desarrollo de la aplicación a instalar en la Raspberry que se encargará de la recopilación de datos, de forma que tengamos el sistema End-to-End final real de extremo a extremo.

. Desarrollos de aviso automático a las autoridades o servicios sanitarios en caso de emergencia.

- Actualización de forma automática de los valores de los sensores que se muestran en la aplicación en vez de utilizar el botón de actualizar datos.

- Mensajería push que envíe notificaciones al dispositivo del usuario para notificar al usuario de un evento.

- Desarrollos para que la aplicación sea capaz de hacer uso de más de ocho campos (por lo tanto más de dos canales) y pueda hacerse una combinación mayor de sensores.

- Parametrizar el nombre de los campos genéricos “Switches Lux”, “Regulador luz”, “Intensidad Luminosa”, “Temperatura”, “Humedad” y “Lluvia” para que desde la propia aplicación se puedan cambiar estos nombres.

Como se puede ver hay muchas cosas que se pueden hacer para mejorar la aplicación. Se ha creado el armazón sobre el cual se debe ir mejorando para crear una aplicación más efectiva y rápida.

2. Presupuesto

El proyecto se puede dividir en:

- Análisis de requerimientos

- Especificaciones

- Diseño y arquitectura

- Aprendizaje de nuevas técnicas de programación

- Programación

- Emulación de los desarrollos

- Documentación

- Mantenimiento

Podemos dividir el proyecto en una primera base de análisis, especificación y diseño que supuso 6 semanas de proyecto. La fase de aprendizaje de nuevas técnicas de programación supuso 4 semanas. La fase de programación fueron 14 semanas. Y la fase de pruebas y documentación fueron 8 semanas.

Las herramientas informáticas que se han utilizado han sido un ordenador de alta gama, con el sistema operativo de Windows 8, y una conexión a Internet, ADSL. En cuanto al personal el proyecto se ha desarrollado por medio de un único programador.

El trabajador ha trabajado a jornada parcial durante los ocho meses de desarrollo de la aplicación.

Por tanto, podemos hacer un desglose de dos tipos de costes. El uso de los equipos junto con el gasto de conexión a Internet y el uso de personal.

Coste del personal

El proyecto ha durado desde el uno de octubre de 2015 al treinta y uno de mayo de 2016. Eso hace un total de ocho meses. Ha habido un trabajo diario de seis horas de lunes a viernes sin realizar ningún tipo de horas extra.

El sueldo por hora de un programador se ha estimado en diez euros netos (se estima hacerlo en neto para simplificar los cálculos de impuestos). A la semana se ha trabajado cinco días a razón de seis horas al día. Lo que hace un total de treinta horas a la semana. En total se han trabajado treinta y dos semanas. Como resultado tenemos un total de novecientas sesenta horas trabajadas. Los cálculos totales se resumen en la siguiente tabla:

	Precio hora neto sin horas extra (10 €/h)
Nº de horas totales trabajadas	960 x 10 = 9.600€

Tabla 2.1. Cálculo del total en euros de las horas trabajadas

Como se puede ver el total de gasto de personal es de 9600€.

Coste de material informático y consumo por uso de Internet

Dentro del gasto de material informático nos encontramos con el uso de los equipos hardware, el uso de licencias software y el consumo de electricidad y conexión a Internet.

Tato el uso de los equipos informáticos como el del software no tiene gasto. Los equipos hardware ya los poseía el desarrollador de la aplicación. El uso de la licencia software de Windows 8 no supone un gasto ya que se presupone como gasto de compra del equipo y por tanto no lo incluimos en el gasto del proyecto.

Lo que sí supone un gasto es la conexión a Internet y la luz. El gasto de Internet lo calculamos mediantes el coste de Internet a la hora y multiplicado por el número de horas trabajadas. Sobre un coste de veintiocho euros con ochenta céntimos por mes lo dividimos por las setecientas veinte horas que tiene de media un mes para saber el precio por hora. Dicho precio es de cuatro céntimos de euro la hora. En total hemos tenido que depender de la conexión a Internet novecientas sesenta horas, lo que nos da un gasto de treinta y ocho euros con cuatro céntimos.

Para el gasto de la luz se estima sobre el precio del kWh contratado y el consumo de los equipos. Se estima que el gasto es de seis euros por las seis horas de trabajo y por equipo. Como se ha trabajado un total de ciento sesenta días el coste total por uso de la electricidad es de mil doscientos ochenta euros.

Los cálculos realizados se resumen en la siguiente tabla

Costes de Internet	38,8 €
Costes de luz	1.280 €

Tabla 2.2. Costes de consumo de Internet y electricidad

Coste global del proyecto

Si sumamos por tanto el coste del personal más el gasto de Internet y el gasto de luz tenemos un coste final de:

Coste final del proyecto	**10.918,8 €**

Tabla 2.3. Coste final del proyecto

3. Manual de usuario e instalación

3.1 Creación de una cuenta en ThingSpeak y ejemplo de uso

Antes de comenzar la descarga del aplicativo HSH_TFG.apk que nos permitirá gestionar los sensores de nuestra casa haciendo uso de una conexión a Internet, debemos realizar los siguientes pasos:

- Registro en la página de ThingSpeak (www.thingspeak.com). Lo primero que debemos hacer es registrarnos en la web de ThingSpeak para de esta forma poder crear un canal donde se empiecen a almacenar los datos capturados por los sensores (Figura 1.2).

- Creación de un nuevo canal y configuración de los parámetros usados en los campos del canal para determinar qué tipo de sensor está asociado a cada campo. Una vez nos hemos registrado procedemos a crear un nuevo canal (Figura 1.3) y rellenamos los campos deseados (Field 1, Field 2, Field 3, Field 4, Field 5, Field 6, Field 7 y/o Field 8) con cualquiera de los siguientes parámetros siempre que no se repita ninguno en dos o más campos:

    - Interruptores de luz: light_s0, light_s1, light_s2, light_s3, light_s4, light_s5, light_s6 y light_s7.

    - Reguladores de luz: light_r0, light_r1, light_r2, light_r3, light_r4, light_r5, light_r6 y light_r7.

    - Sensores de intensidad luminosa: light_i0, light_i1, light_i2, light_i3, light_i4, light_i5, light_i6 y light_i7.

    - Sensores de temperatura: temp0, temp1, temp2, temp3, temp4, temp5, temp6 y temp7.

    - Sensores de humedad: hum0, hum1, hum2, hum3, hum4, hum5, hum6 y hum7.

    - Sensores de lluvia: rain0, rain1, rain2, rain3, rain4, rain5, rain6 y rain7.

Para el caso de ejemplo hemos utilizado los parámetros ligth_s0, light_s1, light_s2, light_r0, light_i0, temp0, hum0 y rain0 en los campos Field 1, Field 7, Field 8, Field 2, Field3, Field 4, Field 5 y Field 6 respectivamente (Figura 1.4 y Figura 1.5). Guardamos el canal con los cambios que hemos introducido.

- Obtención de los parámetros Channel ID, Write API Key y Read API Keys del entorno de ThingSpeak. Una vez hemos completado el paso de la creación del canal y la asociación de los parámetros usados en los campos del canal para determinar qué tipo de sensor está asociado a cada campo, lo que tenemos que hacer es obtener de la web de ThingSpeak los parámetros Channel ID, Write API Key y Read API Keys desde la pestaña API Keys que se encuentra dentro del canal creado (Figura 1.6). Apuntamos el valor de los tres parámetros para su uso posterior a la hora de configurar la aplicación.

3.2 Instalación de la aplicación HSH_TFG.apk y ejemplo de uso

Una vez realizado el proceso de creación de la cuenta de ThingSpeak y de configuración de un canal podemos proceder a realizar la descarga de la aplicación desde el siguiente repositorio de Google Drive:

- Repositorio para descargar la versión 4.2 de la aplicación "Home Smart Home" con nombre el nombre HSH_TFG.apk: Link de descarga del repositorio de Google Drive

A continuación los pasos a seguir para el uso de la aplicación son:

- Instalación de la aplicación tras la descarga del aplicativo desde repositorio.

- Configuración de los parámetros Channel ID, Write API Key y Read API Keys dentro de la aplicación. En la sección de configuración debemos introducir los valores de los tres parámetros Channel ID, Write API Key y Read API Keys (Figura 1.6). En nuestro caso estos tres valores son:

    - Channel ID: En el ejemplo 71882.

    - Write API Key: En el ejemplo BBVUUT4GJO21D5C2.

    - Read API Keys: En el ejemplo FYU7H1REB5VG7LI2.

Una vez hayamos configurados los tres parámetros dentro de la aplicación ya podremos comenzar a utilizar la aplicación. Visualizaremos el estado de los interruptores de luz, reguladores de luz, y los sensores de intensidad luminosa, temperatura, humedad y lluvia. Además podremos interactuar desde la aplicación para apagar o encender los interruptores de luz y modificar el valor de los reguladores de luz.

4. Bibliografía

[1] www.w3schools.com

[2] www.stackoverflow.com

[3] www.desarrolloweb.com

[4] www.getbootstrap.com

[5] www.angularjs.org

[6] www.jquery.com

[7] www.thingspeak.com

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá